DOCUMENT RESUME

ED 231 349	IR 010 757

AUTHOR	Smith, James, Ed.; Moum, Glenda Schuster
TITLE	Proceedings of NECC-82. National Educational Computing Conference, 1982 (Kansas City, Missouri, June 27-30, 1982).
INSTITUTION	Missouri Univ., Columbia.
PUB DATE	Jun 82
NOTE	420p.
PUB TYPE	Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - General (140)
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Literacy; *Computer Managed Instruction; Computer Oriented Programs; *Computer Science Education; Curriculum Development; Elementary Secondary Education; Engineering Education; Higher Education; Mathematics Instruction; *Microcomputers; Programing; Science Instruction; Teacher Education
IDENTIFIERS	Authoring Aids; Computer Games; *Computer Uses in Education

ABSTRACT

Instructional uses of computers with primary through higher education students is the focus of this extensive collection of 61 papers. The papers are categorized by the subject areas of computer-based education, computer science, the computer as a tool in undergraduate education and beyond, computer services, engineering, mathematics, physics, pre-college instruction, social science teacher training, and pre-college computer science. Abstracts and names of presentors are also included for conference tutorials, project presentations, and special sessions which addressed the aditional topics of computer literacy, basic skills improvement, computers in undergraduate education, software applications in elementary and high schools, computer awareness projects, ADA language, and microcomputers for teachers and managers. References are included with some papers and subject and author indexes complete the document. (LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ERIC READING ROOM
127 HUNTINGTON HALL
SYRACUSE UNIVERSITY

Proceedings of NECC–82
National Educational Computing Conference 1982

ED231349

Edited by
James Smith
Glenda Schuster Moum

Hosted by
University of Missouri-Columbia
Columbia, Missouri

Held at
Radisson Muehlebach Hotel
Kansas City, Missouri

June 27-30, 1982

IR010757

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY
Diana Harris

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Copyright © 1982
The Curators of the University of Missouri

Published by: The University of Missouri
344 Hearnes Building
Columbia, Missouri 65211

For: National Education Computing Conference
June 1982

Cover Design by: Cathy Chinn
University of Missouri

NECC-82 Steering Committee

- Richard Austing
  - University of Maryland
- Alfred Bork
  - University of California, Irvine
- Richard Close
  - U.S. Coast Guard Academy
- Nell Dale
  - The University of Texas at Austin
- Karen Duncan
  - ADCIS, HEN
- Gerald L. Engel
  - Christopher Newport College
- John Hamblen
  - National Bureau of Standards
- Diana Harris
  - University of Iowa
- Harry Hedges
  - Michigan State University
- Paul Heller
  - EDUCOM/EDUNET
- Lawrence Jehn
  - University of Dayton
- Sister Mary Kenneth Keller
  - Clarke College
- Jesse C. Lewis
  - Jackson State University
- Doris K. Lidtke
  - Towson State University
- Elizabeth Little
  - Swarthmore College
- Carmel J. Mazzocco
  - University of Missouri-Columbia
- Mike Mulder
  - ESI, Inc.
- Richard Pogue
  - Medical College of Georgia
- Jim Poirot
  - North Texas State University
- Joseph Raben
  - Queens College/CUNY
- Freddy Randolph
  - University of Missouri-Columbia
- Jon Rickman
  - Northwest Missouri State University
- Nancy Roberts
  - Lesley College
- Alan L. Roecks
  - Education Service Center
- Jean Rogers
  - University of Oregon
- Eric Sandgren
  - University of Missouri-Columbia
- Ted Sjoerdsma
  - University of Iowa
- Dennis Spuck
  - University of Houston
- E.M. Staman
  - Systems and Computer Technology, Inc.
- David Stonehill
  - University of Rochester
- Joe Turner
  - Clemson University

NECC-82 Program Planning Committee

GENERAL CHAIRMAN
E. Michael Staman
Systems and Computer Technology, Inc.

CONTRIBUTED PAPERS
Gerald Engel
Christopher Newport College

SPECIAL SESSIONS AND TUTORIALS
Richard H. Austing
University of Maryland

PROJECT PRESENTATIONS
Nancy Roberts
Lesley College

WORKSHOPS
Alfred Bork
University of California, Irvine

and

Eric Sandgren
University of Missouri-Columbia

EXHIBITS
Jon Rickman
Northwest Missouri State University

PUBLICITY & "BIRDS-OF-A-FEATHER"
Carmel J. Mazzocco
University of Missouri-Columbia

LOCAL ARRANGEMENTS
Freddy Randolph
University of Missouri-Columbia

PROCEEDINGS
Glenda Schuster Moum and
James J. Smith
University of Missouri-Columbia

EVALUATION OF NECC-81
Richard Close
U.S. Coast Guard Academy

NECC-82 Exhibitors

The following vendors have reserved exhibit booths for the 1982 conference. Their support is greatly appreciated.

Addison-Wesley Publishing Company, Inc.
Allen Patrick Gray
Anaheim Publishing Company
Apple Computer Incorporated
Aquinas College
Atari, Incorporated
Classroom Computer News
Commodore Business Machines
COMpress
Computer Science Press, Inc.
CONDUIT
Control Data Corporation
DataPhase Systems, Inc.
Developmental Learning Material
Digital Equipment Corporation
dilithium Press
Educational Testing Service
EDUNET
Hartley Courseware, Inc.
Hayden Book Company
Holt Rinehart and Winston
Honeywell Information Systems
IBM
ICCE
John Wiley and Sons, Inc.
Journal of Association for Computers
  in Mathematics and Science Teaching
Loonam
McGraw-Hill Book Company
Mountain Computer, Inc.
Northwest Missouri State University
Output, Inc.
Radio Shack
Random House
Scholastic Inc./Electronic Learning
Science Research Associates, Inc.
Scott, Foresman Electronic Publishing Division
Scott Instruments
Sterling Swift Publishing Company
Syntauri Corporation
T & W Systems, Inc.
Technico, Incorporated
Tektronix, Incorporated
TERAK Corporation
Texas Instruments
The City College of CUNY
The University of Utah
Total Data Systems
SSC/A Houghton Mifflin Company
University of California
University of Missouri-Kansas City
Wadsworth International Group

NECC-82 Sponsorship

The conference is sponsored by the University of Missouri-Columbia College of Education and the following:

- Special Interest Groups of the Association for Computing Machinery:
  - Computers and Society (SIGCAS)
  - Computer Science Education (SIGCSE)
  - Computer Uses in Education (SIGCUE)
  - University & College Computing Services (SIGUCCS)
- Association for the Development of Computer Based Instructional Systems (ADCIS)
- Association for Educational Data Systems (AEDS)
- American Educational Research ASsociation/Special Interest Group on Computer Aided Instruction (AERA/SIGCAI)
- AFIPS Education Committee
- Association for Computers and the Humanities (ACH)
- Association for Small Computer Users in Education (ASCUE)
- American Society for Engineering Education/Computers in Education (ASEE/CoEd)
- Conference on Computers and Undergraduate Curriculum (CCUC)
- Educational Computing at Minority Institutions (ECMI)
- Health Education Network (HEN)
- International Council for Computers in Education (ICCE)
- EDUCOM/EDUNET
- IEEE Computer Society
- Society for Computer Medicine
- Society for Computer Simulation

NECC-82 Paper Referees

Kenneth Abernethy, Virginia Military Institute
Richard Austing, University of Maryland
R.P. Banaugh, University of Montana
Bruce H. Barnes, National Science Foundation
George M. Bass, College of William & Mary
Joyce Baucom, Southern University
James D. Beck, Virginia State University
Della Bonnette, University of Southwestern Louisiana
Alfred Bork, University of California, Irvine
Tom Cain, University of Pittsburgh
N. John Castellan, Indiana University
Sylvia Charp, School District of Philadelphia
Frank M. Clark, Georgia Southern College
Richard Close, U.S. Coast Guard Academy
Ron Cody, Rutgers Medical School
Ronald Collins, Eastern Michigan University
Frank Connolly, American University
M.K. Cueman, General Electric
Nell B. Dale, University of Texas
John F. Dalphin, Indiana University Purdue University at Fort Wayne
Elizabeth-Anne Daly, Christopher Newport College
Herbert Dershem, Hope College
William Dorm, University of Denver
James Eagle, Christopher Newport College
Gerald Engel, Christopher Newport College
Selby Evans, Texas Christian University
F.T. Fink, Michigan State University
David Game, Christopher Newport College
Norman Gibbs, Arizona State University
Fred Gluckson, Advanced Systems, Inc.
Sheldon P. Gordon, Suffolk County Community College
Peg Guyder, Skidmore College
John Hamblen, National Bureau of Standards
Harry Hedges, Michigan State University
Grace Hertlein, California State University, Chico
Larry Jehn, University of Dayton
Dale Johnson, University of Tulsa
Sherryl Johnson, Carnegie-Mellon University
Vincent H. Jones, Southern University
Richard Kerns, East Carolina University
Dan Klassen, St. Olaf College
Marvin S. Koonts, Fairfax County Public Schools
Barbara Kurshan, Hollins College
Doris K. Lidtke, Towson State University
Joyce C. Little, Towson State University
Richard W. Lott, Bentley College
David E. Maharry, Wabash College
J. Mayer, Lebanon Valley College
Sister Joseph K. McAdams, Diocese of Harrisburg
Donald H. McClain, University of Iowa
John McGregor, Christopher Newport College
Edmund D. Meyers, Jr., Boys Town
Andrew Molnar, National Science Foundation
Robert D. Montgomery, North Carolina Central University
Michael Moshell, University of Tennessee
Michael Muler, Electro Scientific Industries, Inc.
Larry Oliver, National Science Foundation
Richard Pogue, Medical College of Georgia
James L. Poirot, North Texas State University
William Porterfield, Hampden-Sydney College
Alice F. Randall, Christopher Newport College

John Randall, Montgomery County Public Schools
Jon Rickman, Northwest Missouri State College
David Rine, Western Illinois University
Nancy Roberts, Lesley College
Alan L. Roecks, Education Service Center - Region 20
Jean Rogers, University of Oregon
Leroy Roquemore,Southern University
William C. Ryan, Swarthmore College
Ted Sjoerdsma, University of Iowa
David Smith, Duke University
Philip F. Spelt, Wabash College
E.M. Staman, Systems and Computer Technology, Inc.
David Stonehill, University of Rochester
Elliot A. Tanis, Hope College
David Thomas, University of Iowa
Robert A. Thomson, University of Dayton
Muralli Varanasi, University of South Florida
Rita Wagstaff, Cuyahoga Community College
Fred W. Weingarten, Information Policy, Inc.
Gary Wittlich, Indiana University
Frank Wojick, VIMS
Peter B. Worland, Gustavus Adolphus College
Allen Ziebur, SUNY at Binghamton

Foreword

The preparation of a forward for a conference proceedings is traditionally among the last ten percent of the tasks required of the general chairman of a conference. As such, it provides an appropriate point in time from which to reflect upon the tremendous effort made by a great number of individuals as they endeavored to bring about the conference. People like to volunteer, and they seem to get a great deal of pleasure from doing the high-quality work that results in an activity such as the National Educational Computing Conference. These are dedicated individuals, whose only payments are the professional recognition of their peers and our gratitude for their efforts. One thing is certain, the Conference would not exist without their help.

Gerald Engel, from Christopher Newport College, had the formidable task of chairing the committee which reviewed some 180 papers and selected 60 to be presented at this meeting. The list of individuals who volunteered and who reviewed the papers appears on the facing page. In addition, Gerry grouped the papers into the sessions which ultimately became the two tracks for presented papers during the Conference. NECC has benefited from his leadership.

Richard Austing, from the University of Maryland, organized and developed both the special sessions section and the tutorials section of the Conference. The tutorials program alone is strong enough to warrant the rating of a "mini-conference" and together the special sessions and tutorials constitute a formidable offering. Nancy Roberts, from Lesley College, again organized the project presentations, a very popular feature of NECC-81. This year the problem of determining how to best group the projects together resulted in a 25% expansion of the projects aspect of the program. Kudos to Nancy for her decisions.

Those who contributed to the extra-curricular aspects of the program also deserve special notice. Al Bork, University of California, Irvine, and Eric Sandgren, University of Missouri-Columbia worked together to coordinate and support the pre-conference workshops. As was the case for NECC-81, this aspect of the Conference has proved to be one of its strong and popular components. Jon Rickman, Northwest Missouri State University, coordinated the exhibits aspect of the Conference. NECC-82 had over three times as many exhibitors participating as in any previous conference, and the success of this effort can be attributed directly to Jon's dedication and hard work.

Carmel J. Mazzocco, University of Missouri-Columbia, was responsible for both the pre-conference publicity and the Birds-of-a-Feather sessions. Freddy Randolph, University of Missouri-

Columbia, was responsible for local arrangements and Richard Close, U.S. Coast Guard Academy, provided the Planning Committee with the evaluation of the previous year's Conference. We owe our thanks to each of these individuals for a job well done.

Finally a special thanks to Glenda Schuster Moum and James J. Smith, both from the University of Missouri-Columbia and who shared the responsibility of editing and preparing the Conference Proceedings. As any who have attempted it will testify, the task of preparing a Proceedings is a particularly time-consuming and arduous one. The quality of their effort is clearly demonstrated in the publication that is now before you.

Finally, thanks goes to all of the indiviuals who chose to attend NECC-82 and who made the Conference a success. As has been stated in NECC's publicity, the purpose of the National Education Computing Conference is to provide a broad and rich forum for discussion among individuals at all levels and from all institutions with an interest in educational computing. Such a discussion is not possible without an active participation on the part of the attendees and, therefore, on behalf of the NECC Steering Committee and the NECC Planning Committee it is to the Conference attendees that these Pro-eedings are dedicated.

E. Michael Staman
Chairman, NECC-82
Systems and Computer
Technology Corporation

Table of Contents

COMPUTER BASED EDUCATION *PAPER PRESENTATION*

1 Determination of Information Paths Among Educational Models
Amanda L. Evans, Toshiyuki Inagaki, David Himmelblau
6 Computer-Based Relativistic Learning
Manard W. Shelly, Lawrence Hoyle, Tamara Whitacre
11 Groups and Computer-Based Learning Materials
David Trowbridge, Alfred Bork

COMPUTER SCIENCE *PAPER PRESENTATION*

15 Computer Curricula in Evolution: A Case Study
Donald W. Kroeber
24 Computer Courses at Hong Kong Polytechnic
N.L. Hickerson, Graham P. Mead
28 Computer-Managed Instruction in the Field of Computer Science
Richard L. Bulinski

PROBLEM TOPICS *SPECIAL SESSION*

37 Academic Computing in Small Colleges: Problems and Solutions
Sister Mary K. Keller, F. Dudley Bryant, Jan Carver, Tom Moberg, Robert A. Rennert
38 Impact of Copyright Laws on Computers in Education
Ron Anderson, LeRoy Finkel, Thomas Gould, Marjorie Kosel

COMPUTER LITERACY *TUTORIAL*

39 Computer Literacy for the Classroom Teacher K-8
Rachelle S. Heller, C. Dianne Martin

BASIC SKILLS IMPROVEMENT *PROJECT PRESENTATION*

40 Using CAI to Teach Adults Basic Skills as a Prelude to Vocational Job Preparation Courses
Howard P. Alvir
40 Keyboarding and Touch Typewriting
Martha Cooper, Mary Jane Sunkel
41 Computer Assisted Instruction in LD Math
Sandra K. Prigger
41 CAI for Developmental Writing: Going Beyond Fill-In Exercises
Michael G. Southwell
42 CAI and Polaroid Interactive Lecture System
Glenn Yoshida

COMPUTER BASED EDUCATION *PAPER PRESENTATION*

43 Interaction in Learning
Alfred Bork
48 Integrating CAI into the Curriculum
Glyn Holmes, Marilyn Kidd
54 The Enhanced CAI Tutorial
Carl B. Jensen

COMPUTER SERVICES *PAPER PRESENTATION*

59 Governance and Policy Regarding Computer Services and Information Systems in Small Two Year Colleges
Lora P. Conrad
65 Development of an Academic Software Management System in a Small College Environmen
Thomas Moberg, Mary Verkade, Thomas Dahl
69 Linking Microcomputers to Timesharing Systems: Efficient Computing in a Liberal Arts Environment
Lynn Arthur Steen

COMPUTERS FOR UNDERGRADUATE EDUCATION *SPECIAL SESSION*

74 Simulation: Real World Applications and Their Implications for Undergraduate Course Content
Nancy Roberts, Richard Bronson, Ralph Deal, Eduardo Riva, Charles Shub

75 Use of Computers in Mathematics: How to Get Started
Sheldon P. Gordon, Raymond Greenwell, Mark Hale, Billie Ann Rice

COMPUTER AS A TOOL IN UNDERGRADUATE CURRICULUM AND BEYOND *PROJECT PRESENTATION*

76 A Computer Information, Analysis, and Simulation System for Teaching in Agriculture
Charles W. Basham, Gail S. Lee, Elizabeth N. Derbyshire
76 Integrating Data Flow Models and System Dynamics in Information Systems Design Courses
Don Booker
77 Word Processing's Usefulness for Writing Instruction
Thomas Flynn
78 Use of Videotape to Present an Introduction to Computing
Susan J. Foster, A. Rees Clark
78 Social Research Simulation on a Microcomputer
William Paisley
79 Development of a Computer-Assisted Tutorial in the Use of SPSS for Application to Health Services Agencies
Melanie Wolf Greenberg

COMPUTER BASED EDUCATION *PAPER PRESENTATION*

81 A Call for the Study of Computer Games
Stewart A. Denenberg
84 Delivery of College Courses Using Educational Technology
Joseph L. Polovney
91 BAFFLES: A New Deductive Game
James D. Spain

COMPUTER SERVICES *PAPER PRESENTATION*

95 User Power: A Discussion and Real-Time Demonstration of Valuable Programming Features
Richard Cornelius
97 A Microlab: Mixing Apples and Oranges
Tom Dooley, Randy Traux
101 The Development of Library Automation in a Medium Sized Academic Library: The U.A.B. Experience
Jerry W. Stephens

SPECIALIZED APPLICATIONS *SPECIAL SESSION*

105 Microprocessors in the Whole of Engineering Education - The Contemporary Micro Laboratory
Thomas A. Brubaker, Tom Cain, John Gowdy, Ronald Krutz
106 Computers for Popular Consumption
Ann W. Lewin, Alan Key, Arthur Luehrmann, Paul Pangora, Ramon Zamora, Blair Sullivan

SOFTWARE DEVELOPMENT *PROJECT PRESENTATION*

107 Development of a Video Computer Courseware Implementation System at the University of Utah
Richard C. Brandt, Barbara H. Knapp
107 ZGRASS: A Simple Language for Graphics Courseware
James Garson
108 Voice Controlled CAI
Carin E. Horn, William D. Wagers
108 A CAI Authoring Environment for Non-Programmers
Frank Rinehart, Lynn Rosenthal
108 A CAI Program for Literary Analysis: MARSY/ERB
Helen J. Schwartz

COMPUTER AS A TOOL IN THE UNDERGRADUATE COLLEGE CURRICULUM & BEYOND *PROJECT PRESENTATION*

110 Modeling Chemical Reactions in PASCAL
Ralph M. Deal
110 Computer Assisted Instruction Project
Margaret Ellis
111 Learning How to Bank at Home
Alice Gagnard, Sara Stone

111 TEST--Test Evaluation and Score Tabulation
Dale A. Schoenenfeld, Dale M. Johnson
111 Microcomputer Facility
Susan Ward, Elaine Reber
112 Using a Microcomputer Network for Instruction of All Ages: Six to Sixty
Nancy Watson, James Watson

EDUCATIONAL SOFTWARE *TUTORIAL*

113 Educational Software: Its Creation and Evaluation
Jerry Johnson

COMPUTER SCIENCE *PAPER PRESENTATION*

114 A More Analytical Approach to Teaching the Introductory Programming Course
Lionel E. Deimel Jr., David V. Moffat
120 A Liberal Arts Computer Literacy Course: Preliminary Report
Stanley P. Franklin
123 Small One User Systems vs. Timesharing Systems in the Introductory Course
Charles M. Shub

ENGINEERING *PAPER PRESENTATION*

129 Low-Level Computer-Assisted Design on an APPLE II with PASCAL
Ralph M. Deal
133 Interactive Microcomputers with Graphics Provide Teaching Assistance in Engineering and Technology
William R. Moore, Robert Baxter
138 A Case Study of an Effective External Evaluation Mechanism
Arthur C. Moeller, Russell J. Niederjohn

TOPICS *PAPER PRESENTATION*

141 The Effect of Technology in the Traditional Afts: Comparison and Contrast of Female and Male Student's Attitudes Toward Ear-Training Computer-Assisted Instruction
E. Margo Dillard, Rosemary N. Killam
155 Computer Literacy: Practical Necessity
Thames F. Jackson Jr., Timothy C. McKee
161 Developing a Computer Curriculum for a College of Education
John F. Vinsonhaler, William D. Wolverton

CURRICULUM APPLICATIONS *SPECIAL SESSION*

165 Implementing Model Curricula
Daniel V. Goulet, John Beidler, Thomas Ho

APPLICATIONS AT ABOUT THE HIGH SCHOOL LEVEL *PROJECT PRESENTATION*

166 Project C.A.I.S.H. - Computer-Assisted Instruction and Support for the Handicapped
Warren R. Brown
167 Microelectronic Technology: Implications for Secondary School Mathematics Curricula
James T. Fey
167 The TOCS (Teacher Oriented Computer System) Approach to Remedial Mathematics
Lloyd A. Jerauld
167 Cross Lingual Mathematics Information System
Jim Kelly
168 Compu Grade: Teachers Gradebook Programs
Robert Kraft, Charles D. Friesen
168 Students Teaching Students About Computers
Alisha Whipple, Charles Grady, Craig Humes
169 The Implementation and Evaluation of Microcomputer Use by Incarcarated Juveniles at the Spofford Juvenile Center
Antonia Stone, Karen Billings
169 The Teachers Desk of Tomorrow, Today
Kara Gae Wilson

COMPUTER SCIENCE *PAPER PRESENTATION*

170 Rating the Major Computing Periodicals on Readability
Ronald S. Lemos

186 An Interactive Instructional CODASYL Data Base Management System for Main Frames and Micros
Marino J. Niccolai, H. Rylan Lewis
192 Using Robots to Help Teach Assembly Language
Jerry Scott

MATHEMATICS *PAPER PRESENTATION*

196 FORTRAN and Introductory Numerical Methods: A Freshman Course
G.C. Borse, C.W. Clump
203 Natural Language Computing in a Linear Algebra Course
Robert Geist, D. Kraines, P. Fink
209 The Use of Computers in a First Course of Numerical Analysis
Zane C. Motteler

PHYSICS *PAPER PRESENTATION*

213 Introductory Quantum Mechanics Using Microcomputers
K.S. Min, Don Kosterman
216 SM 1802 A Microcomputer Simulator
Carl Schmiedekemp
225 Interfacing with ATARI 800 Paddle Ports
Kenneth E. Williams, Michael F. McInerney

MICROCOMPUTERS FOR TEACHERS AND MANAGERS *SPECIAL SESSION*

229 Instructional MIS: User and Management Considerations
Robin D. Avery
230 Teacher Training Programs: Use of Microcomputers
Forrest L. Coltharp, Elton Beougher, Mary K. Corbitt, Richard T. Driver, John J. Rezac

APPLICATIONS IN THE UNDERGRADUATE SCIENCE CURRICULUM *PROJECT PRESENTATION*

231 Computerized Data Logging
Mark Adams, Todd Clonninger, Victoria Collins, Donald Collins
231 Microcomputers and Large Screen Projection in Science Lecture Halls
Richard Cornelius
231 The Molecules in Your Body: A Hard Core Biochemistry Microcomputer Assisted Instruction Molecule
John M. Franz
232 Barriers to the Use of Instructional Computing in the Teaching of Undergraduate Chemistry
Helen H. Nuefeld
232 SUNY Brockport Science and Mathematics Microcomputer Project
Theron Rockhill
233 The SUMIT Courseware Development Project
James D. Spain

COMPUTERS IN MANAGEMENT *SPECIAL SESSION*

234 Electronic Mail
Paul Heller
234 Electronical Conferencing
John Hogan

COMPUTER SCIENCE *PAPER PRESENTATION*

235 Computer Literacy in the Elementary School: A District-Wide Program
Bobby Goodson
238 Computer Literacy 1982: An Update
Grace C. Hertlein
245 A Special Computer for Teaching How Computers Work
John Monk

MATHEMATICS *PAPER PRESENTATION*

248 Programmable Calculators in the MSE Curriculum: A Report on Experiences in the Trenches
James W. McNulty, John L. Palmer
254 Helping Teachers Teach Elementary Mathematics with Calculator, Microcomputers and Manipulatives
Margaret L. Moore

258 The Effectiveness of Mathematics Instruction in the CAI Mode: A Comparative Study
B.M. Pulat, Chaing Wang

PROJECTS IN TEACHING *SPECIAL SESSION*

263 Alternative Teaching Techniques: A Possible Solution to the Crisis in Computer Science Education
Nell Dale, Hal Berghel, Henry Etlinger, Stephen Franklin, Billie Ann Rice, Adele Seeff
264 Microcomputer Education for Teachers: Threats, Techniques and Triumphs
Herb Nickels, Ron Adams, George Culp

UNDERGRADUATE NURSING CURRICULUM USE OF COMPUTERS *PROJECT PRESENTATION*

265 An Authoring System for Developing Computer-Assisted Instructional Programs on Nursing Process
Sue Grobe
265 Course Description for EDUC 572, Readings in Education
Ruth M. Neil
266 The Role of the CAI Author as Curriculum Developer: What Instructional Design Theory Can Tell Us About Planning CAI
John R. Tharp
266 An Orientation to Computer Literacy in Nursing
Barbara Thomas

ELEMENTARY SCHOOL USE OF COMPUTERS *PROJECT PRESENTATION*

268 Reading Skills Courseware Series
Sandra Y. Belton
268 Case Studies of Children Learning LOGO
George Burns
268 The Role of the Classroom Teacher
Michael Cook
269 Teaching LOGO in Child-Centered Classes
Barbara Dubitsky
269 Matching Computer Based Reading Instruction to PreSchool Gifted Children's Modality Strengths: A Whole Brained Approach to Learning
Marcia L. Krause
269 A Humane Forgiving System for Beginning Programers in BASIC
Sister Barbara Kushan
269 Demonstration, Field Test Results, and Discussion of Computer Software Based on Video Game Formats
Marilyn Ruggles, Jerry Chaffin

COMPUTERS IN MUSIC *SPECIAL SESSION*

271 Language and Computer Instruction in the Music Environment
Joseph Raben, Michael Arneson, Mary Dee Harris Fosberg, Dirk L.C. Geens, Edward Purcell

PRE-COLLEGE INSTRUCTION *PAPER PRESENTATION*

272 Computerized Individual Instructional Program
James M. Caccano, Robert W. Watkins
279 Microcomputers and Maxi-Computers: Can We Coexist in Educational Computing?
LeRoy Finkel
285 A Report on Computing in Iowa High Schools
Ted Sjoerdsma

SOCIAL SCIENCE *PAPER PRESENTATION*

289 Teaching Substantive Sociology Through Computer-Assisted Homework: The Development of CLEAR
George H. Conklin
302 STATAN: A Query Language to Help Students Practice Statistics
Anne M. Parkhurst
306 An Empirical Analysis of "learnit", A Microcomputer Program for Facilitating Memorization
Gary L. Schilmoeller, Kathryn J. Schilmoeller, Dennis R. King

SPECIAL ISSUES *SPECIAL SESSION*

311 Education for Careers in Health Computing
Karen A. Duncan, Richard H. Austing, Richard E. Pogue
312 Computers in Public School Classrooms: Teacher Training Issue and Concerns
Sandy Pratcher, Karen Billings, George Culp, Arthur Luehrmann, Jim Poirot

COMPUTER AWARENESS PROJECTS *PROJECT PRESENTATION*

313 Problem Instruction for Microcomputers
Petra Lucia Arrieche
313 Electronic Information Officer
Matilda Butler
314 A Computer Elective Course for Seventh and Eight Graders
Rowena Hacker
314 SDIP Activity III
Maurice Mitchell, Cynthia Lunning, Douglas Carr
315 Computer Literacy in a Small Liberal Arts College
William Schnippert
315 Your Future with Computers: A Career Discovery Workshop for High School Students and Their Parents
Susan M. Zglicznynski

ADA LANGUAGE *TUTORIAL*

317 ADA Programming Language
Robert Mathis

TEACHER TRAINING *PAPER SESSION*

318 What's an Appropriate Way to Teach Educators About the Use of Computers?
Lincoln Fetcher
323 Computer Literacy for Teachers
Judith A. Hopper, Glenn Bruckhart
326 Computer Literacy for Teachers: An Intensive Program
Edward L. Vockell, Robert Rivers

PRE-COLLEGE COMPUTER SCIENCE *PAPER SESSION*

331 Problem Solving Research and Structured Programming in High School
W.E. Baird
335 Algorithms in High School
Josef Gruska
341 An Original High School Programming Course that Works
Bill Shirley

SPECIAL APPLICATIONS *SPECIAL SESSION*

345 Educators for Computing and Health Care - A New Profession
Karen A. Duncan, Sidney Katz, Richard E.Pogue
346 Organization Support for Professionals Interested in Precollege Education
Dick Ricketts, James M. Adams, Jr., Bobby Goodson, Dave Moursund, Larry K. Smith, Dennis W. Spuck

PRECOLLEGE TEACHER EDUCATION *PROJECT PRESENTATION*

347 A Computer Education Course for Elementary Teachers
Philip East
347 Topics and Programs for Elementary and Secondary School Admin.
Ted Mins
348 An Inductive Approach to Learning CAI Programming in BASIC
Tillman J. Ragan, Larry Dillingham
348 Microcomputer Usage for Special Educators and School Counselors, Development of a Symposium
Rosalie Rhoads
349 A Master's and CAGS Degree Program: Computers in Education
Nancy Roberts
349 Overview of Two Mini Courses and Two Conferences in Educational Applications of Microcomputers for In-Service Teachers
Robert Underhill

TEACHER TRAINING *PAPER PRESENTATION*

351 Summer Institutes in Microcomputing for Secondary Teachers
John M. Baker
355 Quality In-Service Training the Key to Establishing and Maintaining a Computer Education Program
Stevan R. Shotwell
359 A Teachers Introduction to Computers
William J. Taffe, Stephen M. Weissman

PRECOLLEGE COMPUTER SCIENCE *PAPER PRESENTATION*

364 The Duke University Computer Kamp, "DUCK"
Kevin Bowyer, Mel Ray, Cary Laxer
369 Computer Education at Moreland Elementary School - The 1st Year
Beth Lazerick
373 A COMAL System for Irish Secondary Schools
Kevin Ryan

SPECIAL APPLICATIONS *SPECIAL SESSION*

379 Faculty Training and Development in Academic Computing
Sheldon P. Gordon
380 Programming Project Management
Rita V. Rodriguez, Herman Acuna, Frank D. Anger, Dr. Babylonia

THE COMPUTER IN THE UNDERGRADUATE MATHEMATICS CURRICULUM *PROJECT PRESENTATION*

381 Calculus Tutorial System
Nicholas Arganski
381 Graphical Explorations
Robert Evans
382 Computer-Aided Education in Calculus -- Limits
Dale T. Hoffman
382 FUNGRAPH: A Language for Computer Inspired Learning
John C. Miller
383 An Undergraduate Curriculum in Computer Graphics
Preston Prigger
383 Laboratory Integration in Finite Math and Survey Calculus
Dale A. Schoenefeld
383 Microcomputers in a Two-College Center for Learning and Instruction in Mathematics: Concepts Building
Noel Evans, Joyce Friske, John Kontogaines, Carla Thompson

Computer Based Education

DETERMINATION OF INFORMATION PATHS AMONG EDUCATIONAL MODULES

Amanda L. Evans
CACHE Corporation
Austin, Texas

Toshiyuki Inagaki
Institute of Electronics and Information Science
University of Tsukuba
Ibaraki-Kin, Japan 305

David M. Himmelblau
Department of Chemical Engineering
The University of Texas at Austin
Austin, TX 78712

ABSTRACT

We describe a method of structural analysis of educational modules that determines the paths to be taken by a student from one or more starting modules to a desired concluding module. Given the pairwise prerequisite information between modules provided by faculty or students, the digraph of the relations is used to isolate the possible paths via computer calculations. Our procedure differs from Interactive Structural Modeling (ISM) procedures in that we can accommodate parallel paths (OR gates) whereas ISM does not.

INTRODUCTION

A number of investigators have described the application of computers to determine the precedence order among courses in a given curriculum (1,2,3,4,5,6,7,8). We focus our attention here on an analogous procedure, namely the determination of paths between pairs of modules in a set of educational modules.

The CHEMI project was initiated in 1979 via a grant to the CACHE Corp. from the National Science Foundation. To date over 400 modules and 500 abstracts of stand-alone, single topic instructional modules have been produced in the core areas of the undergraduate Chemical Engineering curriculum. The modules were prepared for remedial study, self-paced courses, or as substitutes for missing topics in a text-book for a lecture course. Dissemination of the modules will be via the computer.

As part of an information system that will accompany the modules, we wanted to include a diagnostic tool for a student in the form of a map showing how to get from one module to another. We were interested in presenting to the student (and instructor) not only the shortest possible path but all paths that were not redundant. Examination of the ISM method indicated it would develop the shortest possible path satisfactorily, but would delete all the other paths from an initiating module to a terminal module. Consequently, we developed, and describe here, a program called LINK that does yield all the paths between pairs of modules, or elements, in any system.

We first describe the ISM procedure and then the LINK program that we have developed. The ISM procedure is employed in a phase of our method to determine relationships among certain key modules at the same hierarchial level.

ISM

Interpretive structual modeling (ISM) (9) is a technique to quantitatively develop structural relations in a system composed of elements. For example, the elements might be educational modules, or courses. The elements presumable can be clearly identified but the relations, or lack thereof, among them may be somewhat obscure. ISM assumes transitivity of the interrelationships and uses a poll of analysts' views on the (perhaps subjective) pairwise relations among the elements to establish the complete relational pattern among the elements. The perceptions of the system analysts, such as teachers, who are experts in the understanding of the system under consideration are fed into an ISM computer program. Each pairwise perception between elements introduces into the computer the existence or the absence of a specified type of relationship, such as one element is a precursor of another, or there is no relationship.

The judgment of the group of experts regarding the pairwise relations is the foundation on which the structure of the system is built. Note that the group consensus process used in ISM is different from other group decision procedures such as the Delphi method. In the latter approach, a group of experts is asked to provide individual opinions, usually on paper. Each expert is then informed of the opinions of others and asked to reconsider his or her own opinions. The cycle is repeated until a consensus is obtained.

ISM is best carried out via an interactive approach requiring each expert to enter decisions into a computer. An expert may wish to see the effect of a decision just made, that is, to see the inferences that would result using the value supplied for one of the pairwise relations. Such close interaction with the structured model being developed enables an expert to improve his or her decision making ability during the course of entering the data into the computer and interacting with other experts.

Once the pairwise relation decisions are entered into the computer, the ISM program sets up logical inferences from the supplied pairwise relationships and constructs a binary matrix of the relations. The computer program then simplifies the matrix into a compact matrix that can be displayed as a multilevel digraph or a table. The digraph is the structural representation of the original system under consideration. From the matrix or digraph a student or instructor can determine the precedence order among the modules.

ISM does consolidate the relation matrix so as to yield only the shortest path between one element (module) and another element (module). Consequently, if alternate paths exist between two modules, only one path will remain after processing the data, and information about the existence of certain modules may not be revealed to a student. To avoid this difficulty, we have developed a diagnostic program called LINK.

LINK

The LINK program was developed to assist in establishing sequences of modules that a student should study in order to reach a pre-specified goal. The sequences are determined from a set of prerequisite relationships among the modules. Often in a curriculum and in our set of modules multiple paths are found leading from a starting module to a termination module, creating a logical OR gate. The ISM program cannot include OR gates in its analysis of prerequisite relationships, only AND gates. LINK can accommodate both.

How LINK Appears to User

At the outset of an interaction with LINK, the user is asked to identify the target module (the module with which he or she wishes to end his or her study sequence). LINK processes the prerequisite relationship data, including alternate paths, and provides the user with one or more module sequences that should be studied to reach the target module. Multiple sequences result when LINK encounters alternate prerequisite paths to a module. Figure 8 shows a typical output for the LINK program.

Users of LINK can read on a CRT display the necessary information as to how to use the program, namely:

1) The kind of information that they must input to the program.

2) How the output of the program can be interpreted.

Interactive users are requested by LINK to input three kinds of data:

1) A code for the set of modules that contains the modules to be studied.

2) The number of the target module.

3) The number of the module that the user has chosen to initiate study. The user is allowed to specify a plural number of initiating modules if desired.

Three kinds of error protective functions have been included in LINK:

1) If a user selects a module set not included in LINK's prerequisite data base, he or she is requested to reenter a correct module set code.

2) If a user identifies a target module that does not exist in the series he or she has chosen, then he or she is requested to reenter a correct module number.

3) If the target module is found to be prerequisite to the initiating module(s), then the program terminates with an error message to the user. This protection is important because users are not expected to have information regarding the exact hierarchical structure (prerequisite structure) among the modules.

LINK has other error protection capabilities of interest to the programmer or system developer who is entering precedence order data.

LINK Methodology

Notation:

k = vertex label (module k)
din(k)=number of incoming edges to vertex k
dout(k)=number of outgoing edges to vertex k

For example in Figure 1, din(k) = 2 and dout(k) = 3.

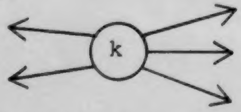


Figure 1 Example of a Vertex and Edges

Definitions:

1) If din(k) $\geq$ 2, then vertex (module) k is called a merging type vertex.

2) If din(k) = 1 or 0, then vertex k is called a non-merging type vertex.

3) If dout(k) $\geq$ 2, then vertex k is called a branching type vertex.

4) If vertex k satisfies any one of the following conditions, then it is called a key vertex:

a) $din(k) \geq 2$ (merging type)

b) $din(k) = 1$ AND $dout(k) \geq 2$ (non-merging and branching type)

c) $dout(k) = 0$ (no outgoing edges).

Consider the directed graph (digraph) in Figure 2 as an example.

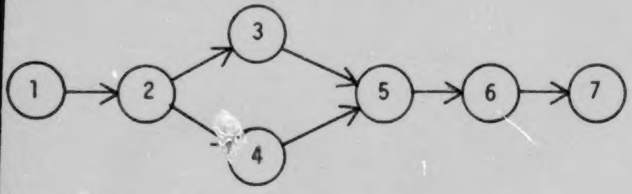


Figure 2 Example Digraph

The key vertices are 2, 5, and 7. The concept of a key vertex is introduced in order to facilitate decomposition of the original digraph into subdigraphs. Decomposition of the digraph in Figure 2 yields Figure 3.

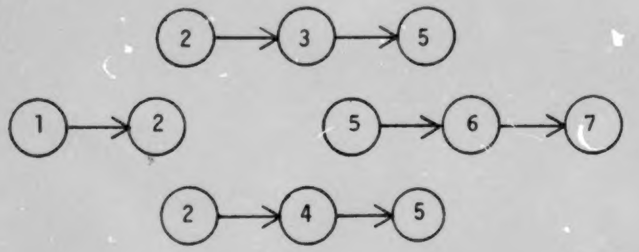


Figure 3 Subdigraphs

The subdigraphs can be classified as shown in Figure 4.

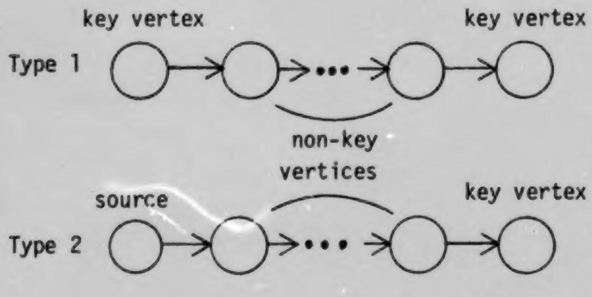


Figure 4 Two Types of Subdigraphs

By doing this kind of decomposition, a unique set of subdigraphs can be obtained. Each subdigraph has only one path from its source (key vertex, or a source in the original digraph) to the sink (the final key vertex in the subdigraph). Hereafter we will call a subdigraph a "subsequence".

Next, a record is prepared for each subsequence. A subsequence of length 1 (consisting of only one vertex with no edges) is introduced for a merging type vertex. Five records are obtained from the subsequences in our example as shown in Figure 5.

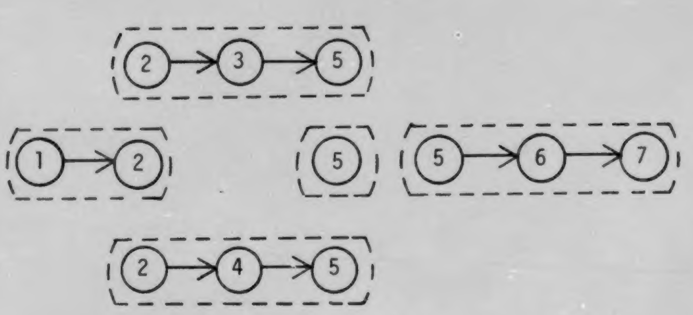


Figure 5 Records Generated from Subdigraphs

The five records are linked by pointers to enable reconstruction of the original digraph. Examine Figure 6.

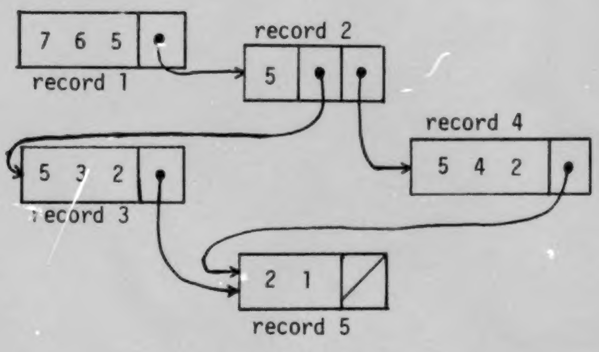


Figure 6 Linked Records

Note the order of vertices in each record. The reversed ordering is adopted for backward searching in the learning sequence from the target module to the initiating module(s).

By decomposing the original digraph into subdigraphs, and by linking records, each of which contains the necessary information regarding a corresponding subsequence, we do not have to use more time consuming matrix operations such as the adjacency matrix or the reachability matrix of the digraph to get the desired precedence order relationships.

EXAMPLE OF USING LINK WITH THE CHEMICAL ENGINEERING MODULES

In preparing a typical set of data for LINK, a digraph (Figure 7) is generated from the appropriate pair-wise relationships between modules. The ISM program is then used to determine hierarchical levels of the key modules. Precedence information is entered into LINK's data files, and the program then can produce study sequences for students on demand. Figure 8 shows the output for the digraph in Figure 7.

Any initiating module or modules may be

chosen, with the provision that they be prerequisites of the target module.

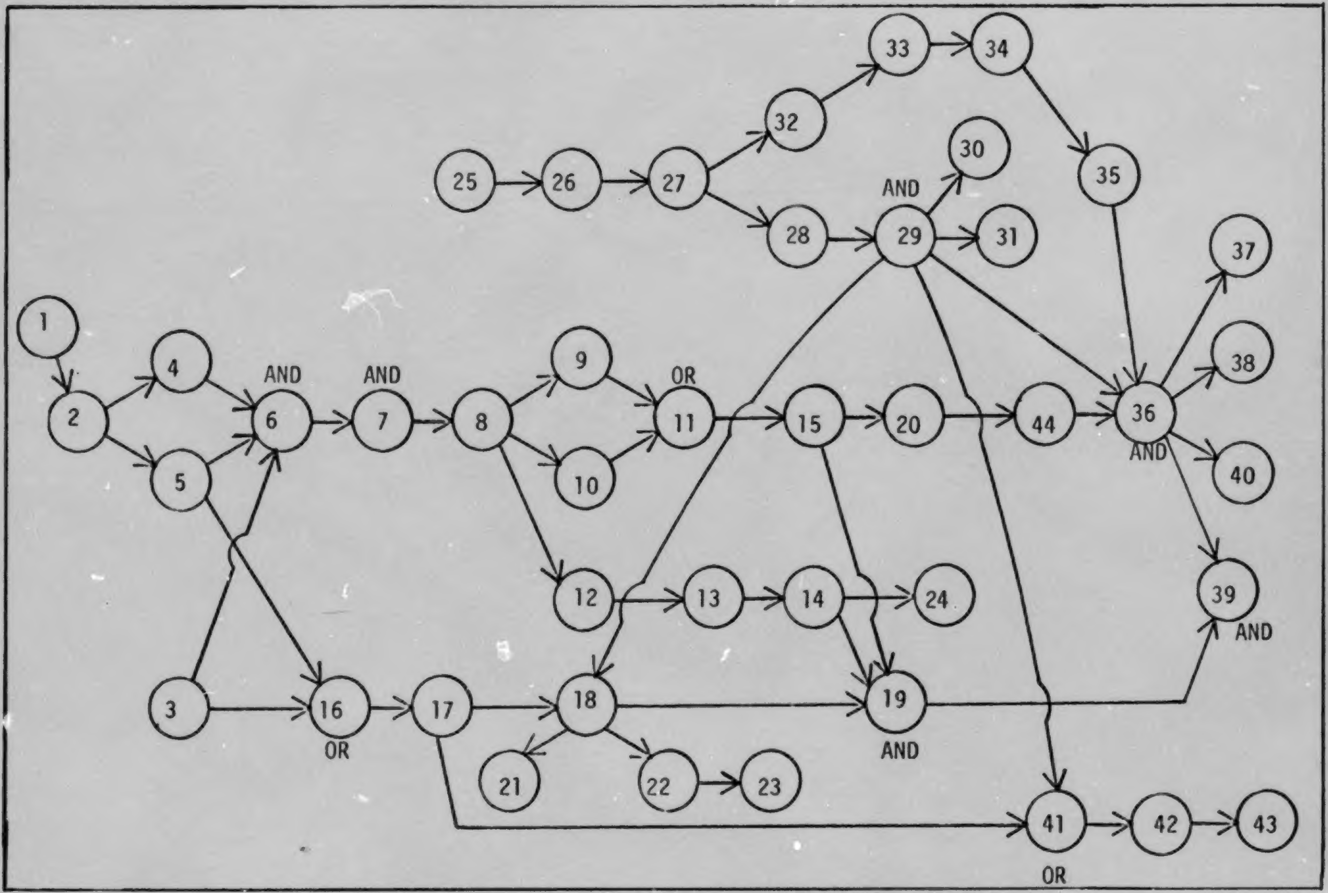

1
2
3
4
5
6
AND
7
AND
8
9
10
11
OR
12
13
14
15
16
OR
17
18
19
AND
20
21
22
23
24
25
26
27
28
29
AND
30
31
32
33
34
35
36
AND
37
38
39
AND
40
41
OR
42
43
44

Figure 7 Digraph of Chemical Engineering Modules

```
MODULE      PREREQUISITES
39          There are 2 sets of prerequisite
            sequences that must be fully finished:
   1)  36 <--
   2)  19 <--
36          There are 3 sets of prerequisite
            sequences that must be fully finished:
   1)  29 <--
   2)  35 <-- 34 <-- 33 <-- 32 <-- 27 <--
   3)  44 <-- 20 <-- 15 <--
29          There are 2 sets of prerequisite
            sequences that must be fully finished:
   1)  28 <-- 27 <--
   2)  18 <--
27     26 <-- 25 <-- 0 <--
19          There are 3 sets of prerequisite
            sequences that must be fully finished:
   1)  15 <--
   2)  14 <--
   3)  18 <--
15     11 <--
14     13 <-- 12 <-- 8 <--
11          There are 2 alternative sequences.
            You must take at least one sequence:
   1)   9 <-- 8 <--
   2)  10 <-- 8 <--

MODULE      PREREQUISITES
 8          7 <--
18         17 <--
17         16 <--
16          There are 2 alternative sequences.
            You must take at least one sequence:
   1)       6 <--
   2)       3 <-- 0 <--
 7          There are 2 sets of prerequisite
            sequences that must be fully finished:
   1)       6 <--
   2)       3 <-- 0 <--
 6          There are 2 sets of prerequisite
            sequences that must be fully finished:
   1)       4 <-- 2 <--
   2)       5 <-- 2 <--
 2          1 <-- 0 <--
```

Figure 8 LINK Output: Recommended Learning Sequence from Module 1 to Module 39 for Digraph in Figure 7

The output of LINK lists each module to be studied, starting with the target module. In Figure 8, module 39 was chosen as the target module, with module 1 as the initiating module. The modules in the study sequence that LINK has generated appear in the left-hand column. The prerequisites for each module are found in the indented rows to the right. In the case of a module with branches as precursors, multiple prerequisite sets are listed.

To establish the paths for study, start with the target module in the left-hand column (module 39). To the right, the message explains that two prerequisite sets are associated with module 39, those ending with 36 and 19, and that they must both be studied in order to be fully prepared for the target module. The user chooses one of the prerequisite sets (say number 1) and looks up the module number in the left-hand column of the list (module 36). For module 36, you find 3 sets of prerequisites. (For module 19 you also find 3 sets of prerequisites). Within a set of modules the dashed arrows show the order in which the modules should be studied. Sequences such as in the second set under module 36, namely 27→32→33→34→35, are an isolated series of prerequisites because the terminating module, No. 35, is not listed in the far left-hand column.

In this way, the user pursues all the prerequisites of the target module until a zero is encountered as the initiating module in a prerequisite set. The zero signifies the end of a prerequisite sequence. Once a zero is reached, the user must return to unpursued prerequisite sets from branching modules (set 2 of module 39 for example), and follow the procedure until all of the sets of modules have been covered.

Note that in some cases alternate paths may be taken, and in such cases only one set of prerequisites _must_ be studied. Examine the comment associated with modules No. 11 and No. 16.

CONCLUSION

In an instructional setting, LINK may be used for curriculum development, student self-help, or as an internal routine running with additional software to provide a user friendly interface, and to possibly make diagnostic decisions between alternate study sequences.

ACKNOWLEDGEMENT

The CHEMI Project is funded by the National Science Foundation under Grant SED-7913021.

REFERENCES

1. Alonso, J.R.F., Henley, E.J., Walton, W., and Finney, R., "Development of demonstration system for automatic generation of block diagrams and fault trees -- Application to curriculum sequencing analysis", Proc. Conference on Computers in Engineering, Canberra, August 23-25, 1978, pp. 115-117.
2. Cohen, K.C., Alonso, J.R.F., and Henley, E.J., "Chemical engineering and modular instruction: A status report", _Chem. Eng. Education_, vol. _12_, No. 3 (1978), pp. 136-139.
3. Heenan, W.A., and Henley, E.J., "Computer aided curriculum analysis", _Chem. Eng. Education_, vol. _11_, No. 2 (1977), pp. 64-67.
4. IEEE Computer Society, Special Issue on "Using computers in education", _Computer_, vol. _13_, No. 7 (1980).
5. Inagaki, T., Henley, E.J., and Koichi, I., "Computer-aided analysis of curriculum structure", _VIII IFAC Triennial World Congress_, Kyoto, Japan, August, 1981.
6. Jedlicka, A., and Meyer, R., "Interpretive Structural Modeling: Cross Cultural Uses", _IEEE Transactions on Systems, Man, and Cybernetics_, vol. _SMC-10_, No. 1 (1980), pp. 49-51.
7. Venkatesan, M., "An alternate approach to transitive coupling in ISM", _IEEE Transactions on Systems, Man, and Cybernetics_, vol. _SMC-9_, No. 3 (1979), pp. 125-130.
8. Waller, R.J., "Comparing and combining structural models of complex systems", _IEEE Transactions on Systems, Man, and Cybernetics_, vol. _SMC-9_, No. 9 (1979), pp. 580-586.
9. Warfield, J.N., _Societal Systems; Planning, Policy, and Complexity_, Wiley-Interscience, 1976.

COMPUTER-BASED
RELATIVISTIC LEARNING

Maynard W. Shelly, Lawrence Hoyle and Tamara Whitacre

Senior author:
Department of Psychology
University of Kansas
Lawrence, Kansas 66045
(913) 864-4126

This paper describes and presents the rationale of a computer-based instruction system for producing relativistic learning. (The meaning of relativistic learning will be explained later). This system, called the ILIA system where ILIA stands for Individual Learning through Interdependent Associations, has been used for approximately two years at the University of Kansas.

THE APPARENT UNAVIODABILITY OF RELATIVISTIC LEARNING

In the earliest times, it seems likely that most formalized learning occurred between a single instructor and one or two students. People were initiated into the arts of hunting and other skills by those who had mastered such skills and who could adapt what they taught to the existing skills and background of the student. As information became more specialized and accumulated in volume, and as the number of persons who could or would teach declined, large classes became more or less standard.

In this envisioned history, the emergence of large classes required a standardization in what was taught. Although in graduate school, and some exclusive academies, among other places, instruction could be individualized to the background and interests of students, in most classes this was no longer a feasible alternative, from grade school through college.

Since standardized material was taught, it was natural to include standardized evaluations as well. This encouraged students to learn what was expected of them even though individualized learning would have been more beneficial. In this view of the history of education, some teachers lost sight of the imperfect relationship between a means of evaluation and an assessment of what had been learned. The possibility that variations in what was learned was desirable seemed to have escaped the awareness of some teachers.

In some areas, of course, a standardization in what is taught is not only desirable, but necessary. The gross structure of man-made systems, such as accounting systems or political systems, are known because they have been created by fiat and can be taught in a standardized way with an emphasis on every student learning essentially the same thing. The same conclusion can be applied to those time-tested analytic schemes that have proved to be successful in the hard sciences and engineering. In these cases, general standardization in what is taught becomes the communication of a scheme of thinking that has been successful against many tests. In the hard sciences and engineering disciplines it usually takes, however, at least five years to acquire and be able to use effectively the standardized information taught.

But in many circumstances the student does not have the background, motivation, or time to understand a complex analytic scheme, or the system may be so complex, as economic systems appear to be, that no generally accepted system for describing it exists. Efforts to teach students useful information about many natural systems suffer from both ailments, the students want easily understood but meaningful answers, and the operation of the natural system is incompletely understood. Even when much is known about some natural systems, as in classical mechanics, the teaching of generally useful standardized information can be difficult because of the motivation and interests of students. The reluctance of many college students to take hard science courses is well known to most professors.

When the complexity to be taught is too difficult for the students, or the system being described is very incompletely understood, two approaches are possible. The first is to teach a standardized set of ideas which necessarily contain inaccuracies and misleading statements. The second is to teach only those aspects of a subject matter which are easy to understand. When the system being taught is poorly understood, the instructor can limit his or her efforts to what is fairly well established.

Both of these approaches encourage the teaching of information that only increases erudition. The material presented may be, and essentially remain, meaningless and hence useless to the student as when a psychologist describes an experiment in detail that interests him or her but has no significance for the students. Such academic hollowisms can be avoided in some cases by the instructor's use of cogent everyday examples making use of shared cultural experiences. This technique is, however, of limited value when meaningfulness depends strongly upon personal experiences with idiosyncratic relationships between memories. In such a case, which is probably more common than uncommon, the learning must become individualized.

The collection of individual memories, associations between these memories, and the implications of these associations upon the way information is processed will be called the individual's mental structure. Information acquired will be called useful information if it influences the individual's mental processes or behavior in a significant way. A complete analytic structure will refer to a largely self-contained intellectual system, many examples of which have already been given, and are epitomized by axiomatic developments in mathematics or physics. When the individual is not being taught a complete analytic structure, and the goal is to teach useful information, then the instructor must make use of the individual's existing mental structure.

Individuals, however, obviously differ considerably in their mental structures. When these differences are relevant to what is being taught, and an effort is being made to teach useful information, individuals will learn different things from the same basic material. Indeed, we feel that it should be the goal of such instruction to teach different individuals different things under these circumstances. In such a situation what each individual does learn, and should learn, is dependent upon his or her mental structure. This is what we mean by relativistic learning. Since complete analytic structures are relatively rare, most useful learning must be relativistic learning.

When psychology is viewed as the study of the entire individual as a natural system, significant learning must be relativistic. This is just one of many possible circumstances in which we feel useful learning in academic settings must be relativistic.

In nonacademic settings, situations requiring relativistic learning are even more numerous. Suppose, for one example, that a manager wishes to learn how to improve the morale in his or her part of a business organization. His or her other responsibilities will limit experimenting with many different alternatives, and the lack of a generally accepted analytic structure will make "looking up" an answer impossible. Additionally, his or her part of the business organization could be quite different from other parts of the same organization, and quite different from similar parts of other companies, making it necessary for the individual to draw upon a relatively unique mental structure.

As a second example, suppose that two FBI agents are trying to track down a particular terrorist group. One agent has an extensive background working on white collar crime. The other agent has had extensive experience tracking gun smugglers. Both agents may use the same fragmentary information to draw different conclusions about the terrorist group. We feel that this is not only natural, given their varying mental structures, but desirable since useful information cannot usually exist without its being integrated into mental structures. The fact that they reached differetn conclusions is indicative of such a relativistic integration. The remainder of this paper is concerned with the following issue: Given the infeasibility of a single instructor relating material to individual mental structures in a large class, can such a process be computerized?

THE MEANING OF RELATIVISTIC LEARNING

We turn now to a more theoretical analysis of relativistic learning. The following are three generally acknowledged features of human mental structures.

1. Associations between ideas differ among people. Freud recognized this as well as nearly every important psychologist and psychiatrist since him. Indeed, psychotherapy may be viewed as an extreme case of relativistic learning. As a simple example, fear for person A may be dominantly related to being robbed on a dark street while fear for person B may be centered upon losing his or her job.

2. The number of ideas connected with a concept or memory will vary among individuals. Frequently this reflects the person's background or experience. A soldier may have more associations connected with the concept of a "gun" than would someone who had never handled one or thought much about weapons. Psychologists have sometimes used such variations in number of associations connected with a concept to reflect its meaningfulness.

3. Forming new associations between concepts or ideas will have different effects for different persons. People often react to the same experience (embodying certain concepts or ideas) differently, as when being a captive produces a life-long linking of hatred of a particular group for one person but a heightened compassion for all people in another person.

Whenever these aspects of a mental structure differ significantly from one person to the next, relativistic learning is virtually unavoidable when not being taught complete analytic structures. Ideas and concepts when not defined largely axiomatically will have essentially unavoidably different meanings for different persons.

Yet, the real significance of relativistic learning may come from constancies rather than from differences. The startling results in the special theory of relativity come from the constancy of the speed of light rather than from inherent differences in varying frames of reference. Similarly the importance of relativistic learning might lie as much or more in what is preserved when learning occurs than in what becomes different or more different.

For the better part of a century psychologists have assumed varying constancies in human mental life. Among the first of such theories was Gestalt psychology, and more recently there is the theory of cognitive dissonance in which some actions are viewed as efforts to maintain a "logical" constancy in what is believed. This latter theory floundered, however, largely because what was to remain constant, "logical consistency," was never adequately defined.

Quantum logic offers a reasonable first approach to defining the constancy that is preserved in mental structures. Unlike classical propositional logic, the most familiar form of logic, quantum logic is essentially an empirical one determined from the findings in quantum mechanics. It is produced by examining the relationships between established conclusions in that area of physics, or more abstractly by calculating the logic of subspaces of a Hilbert space (1). If quantum logic is not itself directly useful in describing mental structures, perhaps a somewhat unlikely event, similar approaches might be applied to mental structures to generate the logic (or logics) appropriate for them.

There are some reasons, however, for believing that quantum logic might be directly relevant to the logic that might be the most important constancy in mental structures. Some writers (2,3) have argued that if one gives "operational meaning" to logical operators (which one might assume must be implicitly done in the logic of mental structures) then one arrives at the quantum logic.

Indeed, both of the above writers have argued for an affinity between quantum logic and mental processes. This possible correspondence between the mind's logic and reality is consistent with Albert Einstein's belief that more mature theories in science are selected less on the basis of external confirmation and more on the internal properties of the theories themselves (4) even though he had a persistent belief that science grasps the real. For him, the inherent "logic" of the mind was extremely powerful.

The use of a quantum-like logic to describe mental processes may also help elucidate elusive but important processes by which individuals learning to deal with a real and complex world. It is well known that without extensive training conformance of mental processes to classical propositional calculus does not exist. Since quantum logic is less stringent (5), it may better describe the ways in which people naturally learn to think about the processes of living. In a highly variable environment, the meaning of probability for individuals has remained a persistent mystery; in quantum logic, probabilities are logical, induced by automorphisms of the logic (6), giving a new perspective on uncertainties. Finally, defining what people mean by entailment, as in theories of personal causation, or attribution theory, has, in spite of these theories, eluded understanding. The difficulties with entailment in quantum logics suggests that in both mental processes and quantum logic new perspectives are needed on issues like perceived causation. Thus, in these and other ways, approaching mental life as preserving a quantum-like logic may give educators new insights into the origins and nature of useful learning.

Thus, as an initial speculation we propose that individuals always attempt to preserve the quantum logic, or some similar logic, of their mental structures, and that this criteria for self-organization can be utilized in developing computerized schemes for relativistic learning.

CRITERIA FOR COMPUTER-BASED RELATIVISTIC LEARNING

We propose, as a first step, that a computerized system designed to enhance relativistic learning should at least meet the following seven criteria.

1. People using the system with appropriate material should feel that they learn more useful information than through non-relativistic instruction.
2. The methods should be enjoyable enough to motivate the student to deal with the complexities intrinsic to useful learning.
3. The methods must be efficient since relativistic learning must often occur in competition with other courses or other responsibilities.
4. Since experience appears to be such an important part in the training of any professional, the instructional program should incorporate real or simulated experiences.
5. Such methods should be usable in many different locales and not dependent upon attendance at a university because of the potentially widespread need for such instructional methods.
6. The methods of instruction should be relatively inexpensive again because of the potentially broad audience they could be addressed to.
7. Finally, of course, the learning should be demonstrably useful in addition to being perceived as useful by the students.

There is evidence that the ILIA system, described below, meets the first six of these criteria. No significant evidence exists on the last crierion.

THE ILIA SYSTEM

The ILIA system has thus far been presented as a conceptual system for teaching relativistic information. It is also an operating computerized system at the University of Kansas, designed, at present, to teach a particular psychology course. It is written in Fortran and Ratfor and operates on the Honeywell 60/66 computer as a batch system when dealing with input from the students and output to the students. The students do not have any direct contact with the computer. It took about 6 years to write the system.

The ILIA system consists of three parts: The source, the method of analysis, and the evaluation techniques. Both the analysis and evaluation are partially dependent upon the source.

The source is information about a real complex system. In the case of the psychology course in which it is used, the source is a 960 question inventory filled out by the student or by someone the student selects. It covers many different aspects of an individual's life. The answers to the inventory along with other information such as scores for later assignments, are placed upon a data base.

The analysis component of ILIA applies the material being taught to the source. In the present circumstance, this component of ILIA applies the concepts of the course to the person filling out the inventory and prints the approximate equivalent of 140 pages of analysis. This printout can be viewed as a personalized text, and is obviously unique for each person. (In the analysis the student is warned against accepting everything in the computer printout as being accurate by repeatedly pointing out that psychology is still an infant science).

The essence of the ILIA system in the present course consists of a series of 42 assignments in which the individual must imagine himself or herself (or the person who filled out the inventory) in particular circumstances, analyze selected situations like a TV program, or actually participate in some simple experiences. Completion of the assignments requires a knowledge of the course material (to understand the questions or to understand relationships upon which the question depends) and of the relationship of the course material to the source. All questions on the assignments are objective, and, obviously, it would be extremely rare for two individuals to have exactly the same "correct answers." (Inventories are checked for duplication). The assignments are scored by comparing the student's answers against the "correct answers" computed from the inventory on the basis of the course material. On each assignment the student receives a score and feedback advising the student about the "errors" that he or she made. There are no tests in the course.

The ILIA system at present works best when students are highly motivated. The complexity of the system (which consists of somewhere between 35,000 and 45,000 lines of program) means that, like a new complex law, it is tempting to look for "loopholes" in order to "beat the system." A considerable amount of effort had to be expended simply to render some guessing strategies nonproductive. This has tended to have a slight negative effect on the realism of the feedback. For example, most answers tend to fall towards the center of most scales because of the usual normal distribution. To eliminate guessing strategies which consistently employed a middle category, "correct" answers involving the extremes of the scales used were weighted much more heavily making statements about the kind of error made more difficult to interpret. (The ILIA system highlights the potential conflict between learning and evaluation).

The concepts introduced in the course are viewed as creating "addresses" which should later be activated by the assignments. A concept like "negative contrast" (the tendency for intense enjoyments to depress later satisfactions)once learned can act as an "address" in a later assignment, such as when the student is asked whether or not an imagined experience might create negative contrast for him or her. Through "activating" this "address" the student can relate the circumstances of the simulated experience to the concepts taught in the course, and thereby relate it as well to other associations aroused by the imagined situation. Since these additional associations will vary from one individual to the next, each student may learn slightly different things from the same question in the assignment.

One inherent difficulty in producing relativistic learning rests on the conflict between evaluation and learning; it is most difficult to evaluate relativistic learning when it is potentially most useful. This is why, for example, it appears that exams in psychology courses are dominated by questions like "A burp is defined as . . .," and "Professor Minutia found that . . . ." From everyday observations and interviews with students it appears that such specific questions do not as effectively stimulate "addresses" as do deeper, more complicated questions. One way in which this problem is partially circumvented in the ILIA system is to use a seriousness index. If the questions in assignments appear to be answered carefully (no obvious guessing patterns are evident), and the student answers most simple questions involving the more basic concepts correctly, "wrong answers" to more complicated questions are effectively discounted. In such a case, it is assumed that the serious student is acquiring relativistic learning which cannot be easily evaluated.

A more direct but more complicated method of encouraging relativistic learning in the ILIA system is under development. Briefly this involves determining some central features of the student's mental structure, and then emphasizing feedback on such features and, in the evaluation of performance, weighting items relevant to these features more heavily. This part of the system may be operational by the fall semester of 1982.

EVALUATION OF THE ILIA SYSTEM

Four evaluations of the ILIA system or parts of it have been carried out, one at the end of each semester for the past two years. Some have focused on accomplishments of the whole system, but most have been concerned with aspects of the system, such as the assignments made, or with the possible dynamics of the learning occurring. These evaluations will be partially summarized in terms of the preceding criteria for assessing a computer-based system designed for relativistic learning.

1. Between 80 and 90 percent of the students report that they felt they learned more useful information from this type of course than the typical psychology course. Some, often psychology majors, felt they learned very little.

2. A slightly higher percentage said that they enjoyed the course more than a typical psychology course, but some disliked the ambiguities in evaluation.

3. Most students consider the course to be fairly easy, which, in view of the first conclusion above, suggests efficient relativistic learning.

4. All assignments involve simulated (imagined) or real experiences. That this is an effective policy rests largely upon informal discussions with students who described the benefits they felt had been realized.

5. The reliance upon batch mode processing for student input and output means that an ILIA system is usable anywhere the mails reach.

6. Excluding disk charges for file storage, the computer costs for using the present ILIA system are relatively small: about $6 for the computer-printed analysis and about 7¢ per assignment for scoring and recording when the computer is used during off hours.

7. No data exists on the "real" usefulness of the course, a characteristic of most psychology courses. About one-third of the students expect the course to have a durable effect upon their lives.

EVALUATION OF THE THEORY

The theory of relativistic learning is not yet explicitly enough developed to be easily evaluatable. Certain findings, however, suggest that it could be a useful theory by leading instructors to focus on previously ignored factors in the acquisition of useful information. Some of these findings are briefly summarized.

"Addresses" which have many connections with other parts of the mental structure should produce more relativistic learning according to the theory. Students who rate themselves as having had a greater variety of experiences than the average person also usually report that they learn more from the course. Informal conversations have suggested that older students (who have had more time to accumulate experiences) obtain more from the course, including two women both over 65.

Meaningful ideas are often those that have more implications for the student and thus have a richer "location" in terms of connections with other parts of his or her mental structure. Students who find the ideas in the course to be meaningful also report that they learn more useful information.

Finally, students who imagine themselves in the situations described in the assignments might be expected to enhance the "addresses" of ideas developed in the course, and thus feel that they learned more from the course. Such an expectation is confirmed.

Finally, interviews with students in the top 10% of the class showed that frequently these students were majors in subjects that required them to interrelate ideas and concepts; such majors were creative writing, journalism, and religion.

CONCLUSION

The nineteenth century gave us the illusion of an objectively understandable world, a universe fully comprehendable by the human mind. Then in the twentieth century the more mature sciences destroyed this illusion. Gödel put an absolute limit on rationality which was later supplemented by other impossibility theorems and computational limits such as those involved NP-completeness. Heisenberg established a limit on what can be observed, and out of a quantum mechanics came a profound duality that stretched and continues to stretch the most flexible of intuitions. And finally Einstein in his theory of relativity demolished the ultimate objective observer.

Yet in the softer sciences, the nineteenth century legacy lives on. Professors in their century old raiments still attempt to teach students with a minimal scientific background useful objective truths about a human being who is many orders more complex than elementary particles, and understanding elementary particles continues to be a task to which many of the Earth's best trained and most astute minds succumb. The need to admit the unavoidability of relativistic learning in domains like psychology is nearly a century overdue.

A recognition of the unavoidability of relativistic learning does not facilitate its accomplishment. It can be expected to be a difficult and obtruse subject whose very nature can be obscured by those psyhcologists and educators who insist upon only specific answers to specific questions. The effective use of computers to produce useful relativistic learning will require a simultaneous appreciation of both the dignity and limits of human beings.

REFERENCES

1. Gudder, S. P. *Stochastic methods in quantum mechanics*. New York: North-Holland, 1979.

2. Finkelstein, D. Matter, space and logic. *Boston studies in the philosophy of science*, Volume 5, 199-215, 1969.

3. Putnam, H. Is logic empirical? *Boston studies in the philosophy of science*, Volume 5, 216-241, 1969.

4. Cooke, R. M. Mystical realism. In Hartkamper, A. and Schmidt, H. J. (Eds.) *Structure and approximation in physical theories*. New York: Plenum, 1981.

5. Bernini, S. Quantum logic as an extension of classical logic. In Beltrametti, E. and van Fraassen, B. C. (Eds.) *Current issues in quantum logic*. New York: Plenum, 1981.

6. Bub, J. What does quantum logic explain? In Beltrametti, E. and van Fraassen, B. C. (Eds.) *Current issues in quantum logic*. New York: Plenum, 1981.

GROUPS AND COMPUTER BASED LEARNING MATERIALS

David Trowbridge
Alfred Bork

Educational Technology Center
University of California
Irvine, CA 92717
(714) 833-5946

INTRODUCTION

Observations of young adolescents suggest that scientific reasoning skills can be enhanced by computer-based learning materials. This paper will describe an investigation to study learning among individuals working in small groups on science activities at the computer.

During field testing of materials produced by the Educational Technology Center, we have observed groups, usually of 2-4 members, form spontaneously and sustain mutually productive learning experiences.

The programs are dialogs on topics in science and mathematics involving a frequent exchange of questions and responses. Considerable use of graphics and some simple animation provide simulations of laboratory experiments, or present a controllable world in which students play the role of scientists.

The dialogs seem to be more effective when: (1) there is a high degree of interactivity with the learning environment, and (2) the materials are used by individuals working together in small groups. The materials produced by the Center were not designed initially for groups; generally, authors envisaged a dialog between themselves and a single student. However, group use seems to be the most common. With a grant from the National Science Foundation,* we began a project to investigate group process as it relates to the use of computer-based educational materials.

The focus of the research has been on: (1) the effectiveness of particular modes of interaction in the computer environment, and (2) the means of collaboration or competition among peers who are involved in a joint learning activity.

First, let us consider the factor of interactivity. Educators generally agree that learning environments which provide a great deal of interaction, such as Socratic dialog and group discussion are superior to ones providing little interaction. While relatively little experience has been gained with computers in the classroom, it is likely that the most effective educational software will provide simulations of hands-on materials and will play the role of an excellent tutor in the inquiry-oriented classroom.

Second, consider learning in groups. Most theories of learning underscore the value of social interaction (Hilgard and Bower, 1975). While there has been a fair amount of research on cognitive behavior among people in small groups, relatively little has been done with computers. When several students work together with computer materials, we see a great deal of questioning, arguing, defending, explaining and helping. They frequently engage in thoughtful discussion, characterized by considerable reflection and insight.

---

*This work is supported in part by NSF RISE grant no. SED-8112633.

BACKGROUND RESEARCH

Relevant research by other investigators falls into two areas: (1) studies of the dynamics of small groups working on learning tasks, and (2) investigations of educational computer applications involving multiple users. This section briefly describes some of this previous work.

In a recent study of young adolescents working on cognitive tasks in small groups, Barnes and Todd (1977) observed that students of average ability, in the proper circumstances, displayed reasoning patterns more advanced than their teachers had expected. When, in the absence of any teacher, these students worked in collaboration on a cognitive task, they often made astute observations and used logical reasoning that was of higher caliber than their typical class contributions.

Frank Murray (1972) investigated the effects of group process on the acquisition of concrete reasoning skills. By grouping two conservers with one non-conserver of nearly the same ages, he found that conservation was induced relatively easily. He concludes, "The data indicate that social conflict or interaction is an important mediator of cognitive growth."

In a study of computer-assisted instruction for college students enrolled in an elementary teaching methods class, Okey and Majer (1976) examined the effect of group size on academic performance. Students worked singly, in pairs, or in groups of 3-4 at PLATO terminals on an instructional module concerning mastery learning.

While no significant differences in cognitive achievement were found, informal observations were of interest. Proctors reported that in groups of three or four, there were occasional instances of incompatibility or domination of a group by one individual, and some reluctance to participate by others in the group. They observed that harmonious and constructive discussion occurred most frequently among pairs of students working together.

The researchers suggest several variables that may affect group instruction at a computer terminal, and recommend further research: degree of active participation, reinforcement of correct responses, novelty, anxiety level, encouragement, and group atmosphere (i.e., competition vs. co-operation; authoritarianism vs. democracy; individual isolation vs. group identification).

RESEARCH DIRECTIONS OF THE CURRENT STUDY

In the present project, we are interested in the acquisition of scientific reasoning skills by young adolescents working together in small groups at the computer. These skills include the ability to:

1. reason with physical concepts involving ratios
2. distinguish between observations and inferences
3. apply operational definitions
4. generate rules from data
5. use evidence to support or refute hypotheses
6. develop scientific models to explain physical phenomena

We are using computer programs already developed at the Educational Technology Center that emphasize scientific and formal reasoning skills. These programs have been described in detail elsewhere (Bork and Kurtz, 1981; Trowbridge and Bork, 1981a,b; von Blum, 1981). Certain dialogs focus specifically on ratio concepts (speed, density), observations and inferences (Sherlock Holmes), operational definitions (area and volume), the generation of rules from experimental data (as in Conway's game of Life, and Mendelian genetics), the testing of hypotheses (laws of light reflection), and the nature of a scientific model (electric current flow).

RESEARCH METHODS

Research of this type requires very detailed observational data. We record behavior by using a video cassette recorder (VCR) with an interface to the computer. A camera focuses on members of the group and a sensitive microphone picks up their voices. We use a recorder with two separate audio tracks, one for sound and the other for keystrokes.

The interface converts digital information read from the keyboard into audio tones that are recorded on the second track. During playback, the VCR drives both an ordinary monitor showing the group, and the computer which recreates the session by simulating actual keystrokes and causing the program to execute in exactly the same way as it did in the original session.

This allows us to examine not only conversation, facial expressions and gestures, but also the computer monitor and echoed input from the keyboard. During playback we can see and hear the learners as well as see what they were seeing on the computer screen and what they were typing. The combination serves as a rich source of information that is used in turn by trained observers to identify and categorize cognitive behavior.

EXAMPLES OF LEARNING BEHAVIOR AMONG GROUP MEMBERS

One of the computer dialogs we tested in a junior high school is called Batteries and Bulbs. Students do a number of experiments with simulated batteries, wires, and flashlight bulbs to discover simple laws of electric circuits and to build a mental model of current flow. One activity deals with a series circuit containing two identical bulbs that glow with equal brightness.

A boy, 11 and a girl, 12 were working on this activity together. They examined two circuits side by side, one containing a single brightly lighted bulb; the other, two dimly lighted bulbs in series. As they tried to understand the fact that in the series arrangement, both bulbs glowed with equal brightness, their conversation went as follows:

B: It matters which way (the electricity) is flowing. If it was flowing this way, then one bulb would have to be brighter than the other.

G: Unh, unh. (Disagrees)

B: Well, if the electricity is flowing this way, ...

G: It's flowing from both ways.

B: Both ways?

G: See, this one (single bulb) is getting all the electricity, and that one (two bulb circuit) is getting it split.

This excerpt illustrates an attempt to consolidate the observation that both bulbs in the series circuit glow equally brightly with the notion that flow would be successively diminished as it passed through each bulb. While the explanation given (that the current flows in both directions simultaneously) is not consistent with the accepted concept of electricity, it does represent an honest attempt to corroborate theory using evidence. Exchanges of this kind seemed to arise quite naturally and frequently among students working in the absence of a teacher.

To illustrate this in a slightly different context, we quote two students who are engaged in an activity called Speed. This program provides simulations of simple rectilinear motions. In one activity, the task is to determine which of two rolling balls has a greater speed when their motions are partially hidden by tunnels.

A white ball and a black ball travel over equal distances; the white ball starts before the black one, and both finish simultaneously. The white ball's speed is less than the black's. When asked the following question, the students respond:

Question: Is the speed of the white ball greater than, less than or the same as the speed of the black ball?

G: Black was faster.

B: No. They were the same.

G: Unh, unh. Because this (black) one started later.

B: I know, but they came out the same.

G: This (black) one started later, so it had to be faster.
Do you want to do it again?

B: Yeah. (Motion is repeated.)

B: (worried tone) I don't get it.
I think it's the same (speed).

In this excerpt, we see a common difficulty of concrete reasoners: in making comparisons, they tend to focus on only one aspect of the situation to the exclusion of all others. Here, the boy focuses only on arrival times in comparing speeds, without taking into account starting times and distances travelled. The girl offers some assistence, but it is not sufficient to enable the boy to discriminate arrival time from speed.

The two examples above illustrate the potential value of cognitive conflict fostered by social interaction. This idea has been suggested by others. In a review of the literature on the acquisition for operational thought, J. Smedslund (quoted by Murray, 1972) states that "the occurrence of communication conflicts is a necessary condition for intellectual decentration."

CONCLUSION

An understanding of group use of computer materials will undoubtedly prove to be valuable in the development of high quality educational software. Furthermore, it will be essential to the evaluation of the effectiveness of this learning medium.

The small group, together with the computer, constitute a system with unique characteristics. A careful description of the human processes that occur in this system will be necessary before the full potential of computer-based education can be realized.

REFERENCES

Barnes, D. and Todd, F. 1977. Communication and Learning in Small Groups, Routledge and Kegan Paul: London.

Bork, A., Kurtz, B., *et al*. 1981. "Science Literacy in the Public Library-Batteries and Bulbs," Proceedings of the National Educational Computing Conference.

Hilgard, E.R. and Bower, G.H. 1975. Theories of Learning, 4th Ed, Prentice-Hall: Englewood Cliffs, New Jersey.

Murray, F.B., 1972. "Acquisition of Conservation through Social Interaction," Developmental Psychology 6(1):1-6.

Okey, J.R. and Majer, K. 1976. "Individual and Small-Group Learning with Computer-Assisted Instruction," AV Communication Review 24(1):79-86.

Trowbridge, D. and Bork, A. 1981a. "Computer Based Learning Modules for Early Adolescence," Computers in Education, R. Lewis and D. Tagg (editors), North-Holland Publishing Company.

Trowbridge, D. and Bork, A. 1981b. "A Computer-Based Dialogue for Developing Mathematical Reasoning of Young Adolescents," Proceedings of the National Educational Computing Conference.

von Blum, R. 1981. "Computers in Informal Learning: A Case Study," Paper presented at ADCIS meeting, March, 1981.

Computer Science

COMPUTER CURRICULA IN EVOLUTION: A CASE STUDY

Donald W. Kroeber

Associate Professor of Management
Department of Management and Marketing
James Madison University
Harrisonburg, Virginia 22801

Phone: (703) 433-2134,6305,6624

ABSTRACT

A number of professional bodies have developed model curricula for computer-related majors at the college and university level. These curricula serve as guidelines for institutions seeking to develop new majors and provide a degree of stability in a rapidly-changing discipline. But institutions of higher education are constrained by available resources, requirements of accrediting associations, the make-up of their student bodies, and, in the case of many public insitutions, their role in a state university system. This paper describes how one institution, James Madison University, developed a comprehensive family of computer-related majors under these conditions over a fifteen-year period.

BACKGROUND

James Madison University (JMU) was established as the State Normal and Industrial School for Women in 1908. The Bachelor of Science degree was authorized in 1916 and the name was changed to Madison College in 1938. Graduate programs were begun in 1954 and in 1966, with an enrollment of 2500, two notable events occurred: men were admitted as full-time students and the first data processing courses were offered by the Business Education Department in the School of Social Sciences. In 1972 a School of Business Administration was formed and in 1977 the present name and university status were adopted. Today, JMU is a comprehensive, regional state university whose 9000 students can choose from among 18 degrees and more than 60 majors at the graduate and undergraduate level.

There are many interesting lessons to be learned from the study of growth in a dynamic institution, but the one of particular interest here concerns the development of computer-related curricula. The introduction of data processing courses into a bookkeeping major in 1967 eventually led to three separate, but complementary majors that span the organizational-technological spectrum. These curricula were developed slowly, without massive infusions of money and other resources, but with the help of a farsighted and supportive administration. Although few institutions today would be forced to follow the same, slow evolutionary path to a computer-related curriculum, the lessons learned at JMU can be applied to the changing environment that surely faces educators in all fields but especially in the high technology disciplines related to computers. A closer look at the development of computer-related curricula at JMU will help to identify these lessons.

THE DATA PROCESSING MAJOR

The data processing (DP) major can be thought of as the "main line" in an evolutionary path that first branched off in an organizational direction with a Management Information Systems (MIS) major and then later spawned a Computer Science (CS) major with a technological emphasis. This evolutionary development is depicted graphically in Figure 1.

The first data processing related major at JMU was actually a joint major in "Bookkeeping-Data Processing" under the Bachelor of Science in Business Education degree, introduced in 1966 in response to a perceived need for secondary education teachers with some familiarity in the growing field of data processing. Only 3 of the 15 courses required for this major were data processing oriented: "Data Processing for Business," "Problems in Data Processing," and "Computers in Business." Punched-Card Machine (PCM) equipment to support these courses was obtained with state vocational education funds and one faculty member, a shorthand instructor, was sent to summer school at American University to become qualified in the subject area. Subsequent faculty were hired against positions generated as a result of a steady increase in student enrollment. Growth in enrollment over the following ten years--at an effective annual rate of 11.8 per cent--was to be the key to introducing new computer-related programs.

The Bookkeeping-Data Processing major remained largely unchanged until 1971 when it took a technological turn with the addition of two courses in programming --"Computer Programming I" (RPG) and "Computer Programming II" (a punched card utility language). By 1973, the punched card course had been more properly redesignated as "Unit Record Data Processing," programming courses in Fortran and Cobol had joined RPG, and the original three DP courses had been combined into a single, introductory course called "Principles of Data Processing." The first academic computer, an IBM 1130, was acquired the same year, again with funds provided by the state for vocational education.

Recognition of the need, outside of secondary education, for college graduates with data processing skills came in 1974 with the introduction (still in the Department of Business Education) of a BS in Office Administration with a major in Data Processing. In addition to a 56-hour business education core curriculum, which included a three-hour "Computer Applications" course, this major required 21 hours of data processing courses: three each in Fortran, Cobol, RPG, PL/1, systems analysis, systems operations, and a seminar in programming.

The DP major took a decidedly organizational orientation over a three-year period beginning 1978 as the School of Business, which had included the Business Education Department from its inception, began to seek American Assembly of Collegiate Schools of Business (AACSB) accreditation. The office administration core of the DP major was gradually changed to the AACSB common body of knowledge. In this period the university also acquired a Hewlett-Packard HP3000 computer to support increasing demands from management, mathematics, psychology, science, education, and, of course, data processing courses.

Although there was no conscious effort to make it so, the DP major turned out to be quite similar to the DPMA model curriculum for undergraduate computer information systems education. (1) A comparison of the current DP major to the DPMA model is made in Table 1.

THE MANAGEMENT INFORMATION SYSTEMS MAJOR

The MIS major, introduced in 1976 before the decision to seek AACSB accreditation had begun to give an organizational orientation to the DP major, was developed in response to a demand for a Bachelor of Business Administration degree with a computer focus. Unlike the DP major, the MIS major was patterned after a recognized curriculum model, in this case, the 1973 Association for Computing Machinery (ACM) undergraduate curriculum in information systems (organizational concentration). (2)

In keeping with the ACM model, the MIS major contained only five "pure" information systems courses with the remainder of the major (three more courses plus the 36-hour BBA core) coming from functional area electives in marketing, finance, accounting, or management. Since Business Education DP courses were cross listed under Management, many students filled out their majors with additional programming or other DP courses, giving a markedly technological flavor to what was intended to be an organizationally oriented major. The five required courses in the major were in information systems concepts (a theoretical approach), advanced MIS (a practical, case study approach), Cobol assembler language programming, and systems analysis.

The use of existing Business Education courses in programming and systems analysis enabled this program to start simply by filling a vacancy in the quantitative stream with a faculty member whose MIS educational and work experience enabled him initially to teach two MIS courses per semester. As the program grew, additional faculty were recruited specifically for MIS.

With some MIS majors virtually duplicating the DP major by taking DP electives and with the DP major moving toward the BBA core to satisfy AACSB requirements, the two programs became uncomfortably similar. There were two obvious alternatives to this problem. One, which was never seriously considered, was to merge the two majors; the other, undertaken unilaterally by the MIS instructors, was to change one of the programs to make it unlike the other. Again, existing resources and personal qualifications of the faculty involved had a more significant influence on the changes than did model curricula.

Feeling that the DP major emphasized routine, administrative applications of business computer systems, the MIS faculty sought to give the MIS major a managerial, decision-making orientation. One step toward achieving this goal was to change the primary programming language from Cobol to Fortran. The decision-making orientation was carried out by requiring majors to select two courses from a series that included courses in operations research, simulation, forecasting, and decision theory (the BBA core already included courses in algebra, calculus, statistics, quantitative methods, and production management). The two MIS courses (theory and application) were retained as were two of the functional electives . The remaining courses in the 27-hour major were a "second programming language," which most majors elect to satisfy with Cobol, and an "additional DP skill," for which a course in data base design is usually recommended. Although it is still possible to take DP courses as functional electives, the quantitative thrust of the major now tends to attract students who are more likely to take the other two of the four available decision-making courses.

Although the changes to the MIS major were made in early 1980 for purely internal reasons, they are very much in line with the revision of the ACM curriculum based on a 1979 survey and announced in November of 1980. This survey found an increased need for emphasis on organizational skills, modeling, and decision making. (3) A comparison of the current JMU MIS major and the revised ACM curriculum in information systems is shown in Table 2.

THE COMPUTER SCIENCE MAJOR

The introduction of a major in computer science in the Fall of 1979 filled the technological gap left by the movement of data processing toward the organizational end of the scale, although this was not the sole motivation for initiating the program. Again, faculty considerations played a large role. At JMU, the Department of Mathematics ranks third (behind Management and Marketing, which houses the MIS major, and Finance and Accounting) in the production of credit hours. But where the two School of Business departments conduct almost 75 percent of their instruction at the upper division level and graduate over 350 majors per year, the Department of Mathematics conducts less than 25 percent of its instruction at the upper division level and graduates only 20 to 30 majors per year. A program that would attract majors would lend balance to Mathematics. Computer science, a major commonly found in mathematics departments when it is not itself a separate department, fit this need rather nicely.

It should not be inferred that the Department of Mathematics sought a computer science program for these reasons alone; the department had a legitimate claim on a technological computer-related major (it already offered courses in Basic, Fortran, and Assembler) and, as noted earlier, such a program complemented the existing DP and MIS majors very well. But, as in the case of DP and MIS, behavioral factors played an important role in the development process.

The CS major follows a model curriculum more closely than either DP or MIS; in this instance, the ACM-78 undergraduate curriculum in computer science. (4) A comparison of the JMU CS major and the ACM model is shown in Table 3.

INTER-MAJOR RELATIONSHIPS

The development of the three computer-related curricula at JMU has been marked by a degree of co-operation not usually found among what might appear on the surface to be competitive programs. Although there was some initial rivalry between the computer science major and the two older School of Business programs, each of the three uses, to varying degrees, the resources of the others. Most obviously, all three share, with other academic programs, common computer facilities. Within the School of Business, the Management Department's MIS major depends heavily on the Business Education Department's programming and other DP courses while the Business Education Department frequently "borrows" qualified faculty from the Management Department to relieve pressure on DP courses. The Mathematics Department, in the College of Letters and Sciences, is less involved in this sharing process although it does accept the Business Education Cobol course as an elective in the CS major and its courses in statistics, Fortran, and Assembler may be substituted for similar courses in the MIS major.

A more subtle form of co-operation is evident at the faculty advisor level where students are apprised of the opportunities that exist in each of the three programs and steered to the most appropriate one without parochial concerns over enrollment figures. This is particularly evident in the computer science program, which tends to attract many freshman whose math skills or career objectives make one of the business programs a better choice. Such students are freely directed to MIS and DP faculty for counseling and advice.

The movement between programs and JMU's rather loose controls over the declaration of a major make it difficult to compare the true program enrollments. Many students who change majors do not bother to make a formal declaration of the change until they are ready to graduate. While this condition casts some suspicion on data concerning the number of majors, the count shown in Table 4 gives a rough idea of the growth and relative popularity of the three programs. The author's personal experience suggests that the total number of computer-related majors for any year is probably close to the correct figure, but many CS majors will ultimately move to MIS or DP.

LIKELY FUTURE DEVELOPMENTS

For reasons unrelated to computer curricula, the Department of Business Education is scheduled to be moved to the School of Education in the Fall of 1983. The move will not include the data processing major, courses, or faculty. While the exact disposition of DP resources and responsibilities has not yet been decided, one option under consideration is the creation of a School of

Business Department of Information Sciences which would assume responsibility for both DP and MIS as well as the quantitative stream which is now emphasized in the MIS major. Perhaps a logical extension of this concept would include the CS major, but that would run counter to the behavioral basis for establishing the CS major in the Mathematics Department in the first place and, thus, is unlikely. Such a combination is not unheard of and may be found in the Department of Computer and Information Sciences of nearby East Tennessee University, a school similar in size, scope, and purpose to JMU.

SUMMARY AND CONCLUSIONS

This paper has been an attempt to present the realities of curriculum development at a regional state university. Many of the administrative details--drafting proposals, winning curriculum committee support, gaining state approval, funding the program, and recruiting qualified faculty--are well known to academicians and have been deleted here, although that is not to imply they were not significant obstacles to overcome. What is perhaps more important to schools seeking to develop computer-related curricula is how programs can be initiated with a minimum of disruption to existing faculty, resource allocation, andprograms.

The JMU experience can be summarized in five points:

1. It is helpful to be in a growth situation. Constant or declining enrollment means that new programs cannot be initiated, or new faculty hired, except at the sacrifice of existing programs or faculty. If the institution is not growing, it must resign itself to very gradual changes facilitated by attrition or to the undesirable tactic of dismissing current faculty.

2. Model curricula are helpful, but should be treated only as guidelines. It is more important to initiate a program within the means of the institution--in terms of faculty, equipment, and other resources--than it is to follow a model. A modest program tailored to the capabilities of the institution has a better chance for success and eventually achieving the goals of the model curriculum than an overly ambitious program that seeks to follow the model from the outset.

3. Traditional department domains should not restrict program planners in developing computer-related curricula. Great resource savings savings can be realized by utilizing existing courses and faculty, whatever their department affiliations may be.

4. Change is essential to a viable program, even if it means a departure from a generally accepted curriculum model. An institution's own faculty is in a better position to assess the needs of the institution and its student body than is the curriculum committee of a professional association, even though the latter may be addressing the needs of the majority better. (This point has obvious limitations, especially when it leads to conflicts with accrediting bodies!)

5. Finally, although traditional liberal arts programs have been on the decline for over a decade, a predominantly liberal arts college can not only survive, but actually experience dramatic growth by adding programs in high-demand disciplines such as data processing, management information systems, and computer science.

END NOTES

1. DPMA Model Curriculum for Undergraduate Computer Information Systems Education, DPMA Education Foundation Committee on Curriculum Development, Park Ridge, Illinois, 1981.

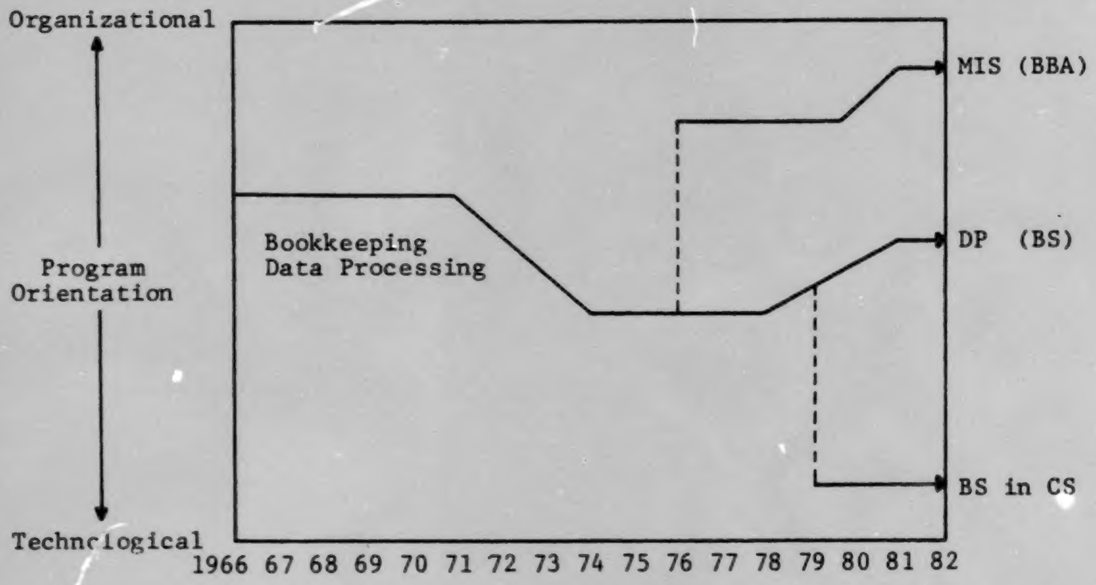
2. Cougher, J. Daniel, "Curriculum Recommendations for Undergraduate Programs in Information Systems," Communications of the ACM Vol. 16, No. 12, (December, 1973).

3. Nunamaker, J.F., "Computing Education in Depth," Computerworld, Vol. XV, No. 39, (September 28, 1981).

4. "Recommendations for the Undergraduate Program in Computer Science: A Report of the ACM Curriculum Committee of the ACM," Communications of the ACM, Vol. 22, No. 3, (March, 1979).

Figure 1

Curriculum Evolution



DPMA CIS Model	JMU DP Major
CIS Core	Major Courses
Intro to Computer-based Systems	Computer Applications*
Appl Prog Dev I (Cobol)	Cobol
Appl Prog Dev II (Cobol)	Advanced Cobol
Systems Analysis Methods	Systems Analysis
Structured Sys Anal & Design	RPG II
Data Base Program Development	Seminar in Programming
Applied Software Dev Project	
CIS Electives (3)	Major Electives (3)
Software & Hardware Concepts	Fortran
Office Automation	Internship in DP
Decision Support Systems	Systems Operations
Advanced Data Base Concepts	Data Base Design & Analysis
Distributed Data Processing	(Functional area elective)
EDP Audit and Controls	
Info Sys Planning	
Info Sys Resource Mgt	
Business Support Courses	AACSB Common Body of Knowledge
Financial Accounting	Financial Accounting
Managerial Accounting	Managerial Accounting
Quantitative Methods	College Algebra
Principles of Management	Calculus
Principles of Finance	Statistics
Organizational Behavior	Quantitative Methods
Production & Opns Mgt	Principles of Management
	Principles of Marketing
	Organizational Behavior
	Micro Economics
	Macro Economics
	Production Management
	Business Law
	Business Policy
General Education Courses	University General Studies
Arts, Sciences, humanities	Arts, natural sciences, social sciences, humanities, English, history, oral communications, P.E.

*Part of AACSB Common Body of Knowledge; shown as part of major for comparison only.

Table 2

A Comparison of MIS Curricula

ACM-78 Info Sys (Revised 1980)	JMU MIS Major
Prerequisites/Func Area Rgmts	AACSB Common Body of Knowledge
Computer Programming	Same as DP Major
Quantitative Methods	(See Table 1)
AACSB Common Body of Knowledge	
Info Sys Concepts & Org	Major Courses
Info Sys in Org	Info Sys Concepts
Information Analysis	Advanced Info Sys
Systems Design Process	Fortran
Info Sys Projects	
Info Sys Technology	Major Electives (6)
Computer Concepts & Software	Decision Tools (Select 2)
Program, File, & Data Structure	Simulation
Data Base Mgt Sys	Operations Research
Data Comm Sys & Networks	Forecasting
Modeling & Decision Making	Decision Theory
	Second Prog Lang (Select 1)
	Cobol
	Assembler
	RPG II
	Add'l DP Skill (Select 1)
	Data Base Design & Analysis
	Advanced Cobol
	Systems Analysis
	Systems Operations
	Seminar in Programming
	Functional Area (Select 2)
	Any upper-level course in management, marketing, finnance, or accounting
	University General Studies
	Same as DP Major
	(See Table 1)

Table 3

A Comparison of Computer Science Curricula

ACM-78 CS Model	JMU CS Major
CS Core	Major Courses
Computer Programming I	Fortran
Computer Programming II	Assembly Language
Intro to Computer Sys	Computer Sys & Architecture
Intro to Computer Org	Adv Prog & Data Structures
Intro to File Processing	Design & Anal of Algorithms
Op Sys & Computer Architecture	Programming Languages
Data Structures & Algorithm Anal	
Organization of Prog Lang	
Electives (4)	Electives (2)
Computers & Society	Numerical Methods
Computer Architecture II	Combinatorics
Data Base Mgt Sys Design	Selected Topics in CS
Artificial Intelligence	Numerical Analysis
Algorithms	
Software Design & Development	
Theory of Prog Lang	
Automata Computability & Formal Lang	
Numerical Analysis	
Linear Algebra	
Special Topics	
Mathematics (5)	Cognate Studies
Intro to Calculus	Analytic Geometry & Calc I
Mathematical Analysis I	Analytic Geometry & Calc II
Mathematical Analysis II	Analytic Geometry & Calc III
Probability	Linear Algebra
Discrete Structures	Probability
Probability & Statistics	Differential Equations
	Abstract Algebra
	Electronic Instrumentation
	University General Studies
	Same as DP Major
	(See Table 1)

Table 4

Program Participation

a. Majors

Year	DP	MIS	CS	Total	% Inc
1977	59	13	-	72	-
1978	94	34	-	128	77.8
1979	120	48	-	168	31.3
1980	130	74	79	283	68.5
1981	148	137	187	472	66.8

b. Graduates

YEAR	DP	MIS	CS	Total	% Inc
1977	?	0	0	?	-
1978	?	3	0	?	-
1979	26	10	0	36	-
1980	26	15	0	41	13.9
1981	27	21	2	60	46.3

Source: James Madison University Office of Institutional Research

COMPUTER COURSES AT HONG KONG POLYTECHNIC

N.L. Hickerson
Head, Computer Centre

Graham P. Mead
Head, Dept. of Computing Studies

Hong Kong Polytechnic
Kowloon
HONG KONG
3-638344

1. INTRODUCTION

This paper contains a description of the courses offered by the Department of Computing Studies at Hong Kong Polytechnic.

The courses are discussed in the light of the environment in which they are offered. The paper begins with a description of Hong Kong Polytechnic, then of the Polytechnic's Computer Centre, followed by an outline of the Hong Kong Computer Industry.

The paper then goes on to describe the Department, the needs which it is attempting to meet, and the courses offered to meet these needs.

A significant aspect of most of the courses is the period which each student spends working in a local DP department. This Industrial Placement period has been very successful, mainly because care is taken to integrate it into the students' program of study. Some discussion of this aspect of the course offerings is discussed later in the paper.

2. HONG KONG POLYTECHNIC

Hong Kong Polytechnic, opened in 1972, is a tertiary level educational institution and, together with the 2 local Universities, is at the apex of the educational system in Hong Kong. It specializes in vocational and professional courses in a wide range of disciplines. In 9 years it has grown to accommodate some 6,500 full-time students and 19,500 part-time students who are studying at various levels of intensity.

There are 20 academic departments ranging from Business Studies and Accountancy in the commercial area to five Engineering departments at the more scientific end of the spectrum. Design, Nautical Studies, and Textiles are some of the other areas of study.

In keeping with the normal practice in British Polytechnics, several course types are offered. Full-time students may complete a Diploma in 2 years, a Higher Diploma in 2 or 3 years depending upon the point of entry, or an Associateship in 1 year following a Higher Diploma. 3 or 4 year Degree courses are being planned.

The courses currently being offered by the Department of Computing Studies are discussed in Section 5 below.

Not only are there a wide variety of courses available to students in the Polytechnic, the modes of study vary significantly. Apart from Full Time students, there are Part Time Evening, Part Time Day Release (normally one day and 2 evenings a week) and 'Sandwich' students, known as co-op students in the United States.

Inevitably the Polytechnic's courses are conditioned by the type of students enrolled. Their ability, background, attitude toward work, and personal aspirations are important factors in their approach to academic work.

Firstly, Chinese students are renowned for the number of academic prizes they win in international competition. At Hong Kong Polytechnic, where only about 10% of the qualified applicants gain places, the academic standard of intake is high by any standards.

Secondly, the students' cultural background results in major differences in attitudes compared to those of western students. On the one hand the respect shown to teachers, and the rather autocratic educational system in Hong Kong, result in a somewhat unquestioning attitude in the acquisition of knowledge. On the other hand, the high value placed on educational achievement is a very positive factor in student motivation.

Thirdly, some 60% of the students are underprivileged by western standards. This is reflected not simply in family income, but equally importantly in the conditions for study at home. Many students live with their families in cramped conditions with little or no opportunity for silence or solitude in which to study.

Fourthly, the majority of students speak Cantonese as their first language. The normal language for teaching at the Polytechnic is English, in which most students are reasonably proficient upon entry. Nevertheless in the Computing Studies Department, in particular, the variety of nationalities amongst lecturing staff, and the corresponding variety of accents, can cause problems. Considerable emphasis is given to English teaching in the computing course, not just to assist during the course itself but rather to prepare for the working environment of which the students will eventually become part.

3. COMPUTER CENTRE

The Polytechnic is fortunate in having one of the most advanced computer centres in the region. Originally it was intended that the 2 Universities and the Polytechnic would share a single machine and the majority of the computer needs would be provided by this means. However trends in technology, and the overheads inherent in managing and planning a central resource for three disparate organizations, subsequently indicated that each institution should have its own computer centre.

The emphasis on computing at the Polytechnic is heavily oriented towards interactive, on-line working. At present some 270 terminals are available for staff and student use, and this number is expected to increase substantially over the next two years.

The rationale for the emphasis on interactive computing has many aspects. Program development is quicker in on-line mode; part-time students, in particular, must have rapid response in order to make progress during the limited time they are on campus. Student motivation, especially amongst those taking computing as a minor subject, is enhanced by hands-on experience. For the given work-mix an interactive approach is cheaper. Finally, and most significantly, such facilities are more appropriate for the local environment since most computer installations in Hong Kong operate in interactive mode.

The Polytechnic has taken a multi-machine approach to hardware provision. Again, the rationale has many aspects. Using multiple medium-sized machines is the only way in which the number of terminals required could be supported within the prevailing cost constraints. Multiple machines allow a rolling update program, thereby avoiding the major upheaval when a single mainframe is replaced. Frequent machine acquisitions allow new technology to be implemented quickly. Lastly, this approach is the most suitable in preparing students for their future work environment.

A further benefit of multiple medium-sized machines only became apparent after the first few had been installed. In Hong Kong, where the manufacturers are many thousands of miles distant, it is sometimes difficult to obtain satisfactory maintenance service for equipment. With multiple machines it is cost-effective to operate an in-house maintenance facility which is more responsive, more expert, and cheaper than any service the vendors provide.

4. HONG KONG COMPUTER INDUSTRY

The first computer was installed in Hong Kong in 1963 and, in terms of machines, the growth has been a fairly steady 30% year after year. However it was not until the early 1970s that the base became large enough for the growth in actual numbers to be significant.

The fact that computerization did not really begin until some 10 years ago is important. When minicomputers became widely available, in the early 1970s, there was no legacy in Hong Kong of big machines and big machine thinking. Thus the new technology and associated ideas (e.g. a fully interactive approach to computerization) were more readily adopted.

The advent of minicomputers was particularly relevant to Hong Kong, where a high proportion of businesses are small or medium sized. Minicomputers provide an affordable solution for the smaller company.

There are today about 700 computer installations of any size. The number of microcomputers is impossible to estimate.

A feature of Hong Kong is that most computing is "commercial" in nature. Very few scientific applications exist at the moment, but this may well change soon.

The level of computerization is mixed by western standards. The most advanced systems are as sophisticated as any in the world, whereas the manufacturing sector, for example, has been slow to change from its traditional methods of operation.

By far the greatest problem facing the Hong Kong computer industry is the shortage of appropriately trained manpower. This is a problem in most countries, but is particularly acute in Hong Kong for a number of reasons. Firstly, unlike in other countries, Hong Kong did not have a pool of expertise on which to draw when the rapid growth in commercial applications began. Secondly, there was no computer education in Hong Kong (apart from short courses) until the Polytechnic enrolled its first computer major in 1975. Apart from a limited number of computer majors at the 2 universities the situation is much the same today. Thirdly, the somewhat mobile nature of the population results in the number of graduates in computing who return having obtained a qualification overseas being more than matched by the emigration of experienced people.

It is in the context of a severe and worsening shortage of qualified personnel and extremely limited educational resources that the Polytechnic is offering its computer courses.

5. THE DEPARTMENT OF COMPUTING STUDIES

In previous sections some of the more important aspects of the Hong Kong situation, the background of the students and the state of the local computer industry, have been discussed. These will be referred to again in this section when considering the work of the Department in its efforts to meet the community's needs.

These needs fall into 4 broad categories, as follows:-

(i) Small to medium sized DP departments

The smaller DP departments have special needs. Typically only 2 or 3 development staff are employed, each of whom has to undertake many jobs, particularly when a colleague is away or leaves. Few such installations undertake much sophisticated work. A broad-based education is required in these circumstances.

(ii) Medium to large sized DP departments

The larger DP departments have sufficient staff to allow more specialization. Frequently the work is of an advanced nature. Consequently the education required is somewhat narrower, but to greater depth.

(iii) Management Services Departments

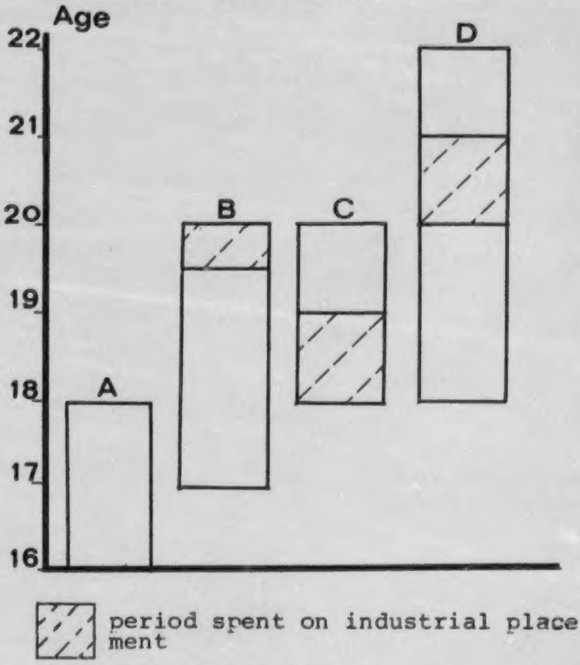
The trend, particularly in the largest organizations, is towards providing a broader service to management than simply the computerization of individual systems. No standard name has developed for the Departments who provide such a service, but 'Management Services' is most common. These Departments are concerned with information flow throughout the whole organization; they view the computer as a tool in the management of information as an overall corporate resource. An appropriate education must reflect this wider view of computer use.

(iv) Career Development of Individuals

The rapidly changing technology and the rapidity of promotion in the computer profession are such that a comprehensive and flexible program must be available to allow experienced computer professionals to keep up to date with current developments.

The Polytechnic addresses each of these areas. A 2-year Diploma in Computing Studies is directed towards the generalist discussed in (i) above; a 3-year Higher Diploma in Systems Analysis is designed for more specialized education, as in (ii); a 4-year Bachelors Degree is planned for the Information Systems Analyst in (iii); while a 2-year Higher Diploma in Computing Studies is offered for those mentioned in (iv) that wish to update their skills.

The course structure, with typical age levels, is shown below:-



Course A - Diploma in Computing Studies
Course B - Higher Diploma in Systems Analysis
Course C - Higher Diploma in Computing Studies
Course D - Bachelors Degree in Computing Studies (planned)

The shaded areas represent periods of time spent on Industrial Placement; that is, time spent working in a DP department on appropriate tasks and supervised both by a company employee and by a Polytechnic lecturer. These periods are vital in producing the sort of practical person which the local industry needs and are the subject of the next section.

All the courses are commercially oriented with substantial emphasis on practical work.

Since English is a second language for almost all students, particular attention is paid to the development of the communication skills, and the confidence to express oneself both orally and in writing, which are so necessary in the computing profession.

The Department is greatly assisted by its Advisory Committee, a group of local DP experts, in determining the syllabus content of its courses, and in other ways.

The Department's courses, as is common throughout the Polytechnic, are heavily oversubscribed. Only 10% or less of qualified applicants are able to gain a place. Since those offered places are chosen primarily on their previous examination performances the academic level of those entering the courses is high.

Staff have been recruited from a variety of countries, and their various backgrounds are useful in providing computer education in a cosmopolitan environment such as Hong Kong.

The students, not surprisingly in view of the shortage of qualified personnel, find jobs easily. Many obtain multiple job offers before completing their course, and the salaries which they command tend to be on the high side.

6. INDUSTRIAL PLACEMENT

One of the most important - and successful - aspects of the Department's courses is the Industrial Placement scheme. It has been in existence for 4 years now, and is considered valuable by academic staff, students, and the companies alike.

The Industrial Placement period is an integral and assessable part of the course. The work which the students undertake is chosen well in advance through discussion between representatives of the Polytechnic and the company, and must be relevant both in content and level of difficulty.

During the Placement period the student is visited on a number of occasions by his Polytechnic supervisor, who also discusses progress with the company supervisor. Reports are completed for each visit.

Should a student encounter any difficulty in his work he may refer the matter to either supervisor as he wishes.

Upon completion of the Placement period the student is assessed. Clearly it is difficult to maintain comparability of assessment between students working on different tasks, in different environments, and with different supervisors. Consequently a pass/fail assessment only is used, with borderline cases being referred to a small group of academic staff charged with the responsibility of judging marginal performances.

The purpose of the Placement period is to relate theory and practice. By judicious choice of project, and through proper guidance, this scheme provides a valuable and rewarding experience whilst giving the practical orientation to the course which the industry requires.

A number of side benefits also accrue. Students are treated, as far as possible, as employees of the company and they come to see how businesses work; the period has a noticeable maturing effect on the students; it gives both company and student a no-commitment view of each other prior to any offer of full-time employment being made; it provides a service to the industry; and it keeps academic staff in touch with the real world.

The Department is fortunate in enjoying strong support from the local computer industry for this scheme. This year, for example, the number of places offered was 3 times the number of students available.

7. CONCLUSION

This paper has described the courses offered by the Department of Computing Studies at Hong Kong Polytechnic and the environment in which they are conducted. Particular emphasis has been placed on the Industrial Placement period because of its importance to the courses and the success which it enjoys.

The courses and the environment are, in some respects, unusual and may be of interest to others who are concerned with developing computer courses in a tertiary level educational institution.

COMPUTER-MANAGED INSTRUCTION IN THE FIELD OF COMPUTER SCIENCE

Richard L. Bulinski
IBM Corporation
P. O. Box 1900
Boulder, CO 80302
447-1900

CONTENTS

ABSTRACT
INTRODUCTION
COURSE DESIGN
COURSE DEVELOPMENT
DESCRIPTIONS
  Exhibit A
  Exhibit B
  Exhibit C
  Exhibit D
  Exhibit E
  Exhibit F
  Exhibit G
  Exhibit H
OTHER AUTHORING/PRESENTATION FACILITIES
SUMMARY
REFERENCES
EXHIBITS A-H

ABSTRACT

The purpose of this paper is to present a course author's view of the design and use of a currently available CAI/CMI software product. The emphasis of this tutorial is on:

- Construction of the pedagogical design of the course material, and
- The mechanics of implementing that design to produce the learning materials presented to students who will take the course.

This paper does not address the issues of formulating the course objectives, identifying the audience level, or validating the effectiveness of the course presentation material.

Illustrative examples used in this paper are taken from a CMI course written to teach budget and financial analysts the use of a software language to assist them in performing Business Planning, Budgeting, and Forecasting analyses. For ease of reference, the acronym "PLANCODE/I" is used to refer to the IBM Program Product, "Planning, Control, and Decision Evaluation System/Interactive". The CMI software used to implement the training course, described here as typical of the course-authoring tools now available, is the IBM Program Product, "Interactive Instructional System". (Technically, the latter is available as two overlapping components, the "Interactive Instructional Authoring System", IIAS, for use in computer-based course materials development, and the "Interactive Instructional Presentation System", IIPS, for use in presenting the resulting courseware to the students.)

INTRODUCTION

The PLANCODE/I Program Product is documented for the novice and beginning user in a "General Information Manual"[1] and a "Program Reference Manual"[2] (PRM). As their names imply, those publications are not suitable for self-study by themselves. Consequently, a "Student Text"[3] was written to reorganize the material of the PRM into learning sequence, adding examples and figures to supplement the enhanced textual matter.

It was apparent that the effectiveness of the Student Text would be improved if student interaction were increased. Yet, there was no valid reason to ignore the useful material that existed in the Student Text and in the PRM by starting the project from scratch. Hence, the decision was made to develop a CMI approach to teaching PLANCODE/I, using the IIAS vehicle to direct the student in his progress through the two existing sources.

As Educators, you are well aware of the problem of attention span in the learning process. That is to say, a video display tube which the student uses in the CAI/CMI manner must be more than a screen full of words and numbers--or less than a full screen, at the least! The terminal must be more than 'an electronic page turner'. Particularly in the CMI environment, the

student:
- Must be able to refer to screens (pages) forward or backward from the present one at will;
- Should be able to view text pages selectively, out of sequence, and from a different perspective than the author's; and
- Should have a non-erasable record of some of the lists, text, and outlines that are part of the course material displayed on the terminal.

For these reasons, a Student Course Workbook was developed as part of the course materials, to accompany the visual displays, and to coordinate the student's progress through selected assignments in the PRM and the Student Text.

COURSE DESIGN

The PLANCODE/I course was designed as a set of 16 "Sessions". (The word 'Session' does not imply that the student must complete a session in its entirety at the terminal before taking a rest break. As used by IIAS and CAI generally, Session is comparable to a Chapter or Module).

Each Session is organized as follows:
- Session Title screen.
- List of the one to four Objectives for and corresponding Topics covered in the current session.
- Verification that the student has completed the Reading Assignment for the current session. Offer to interrupt if not.
- Offer to the student to elect to bypass the session, if already familiar with the material.
- The "branching" logic FOR EACH TOPIC is as follows:
  - Pre-test question, in completion, multiple choice, true-false, or matching or ranking types.
  - Tutorial text, displayed only if the student missed the correct answer to the pre-test question.
  - Post-test question, displayed only if the student was shown the tutorial text.
  - Review/Remedial text, displayed only if the student missed the correct answer to the post-test question.
- Summary of the entire Session, and offer to the student to elect to review any one or more of the topics of that session.

The first session also contains an orientation to the mechanics of and facilities for taking the course, such as:
- Statement of overall course objectives and intended audience.
- Session subjects list.
- Required course materials.
- Student actions:
  - Sending comments/questions to the local course Administrator.
  - Requesting hints to the meaning of a question.
  - Requesting display of the definition for a technical term. (Compiling and entering an on-line Glossary is an author responsibility, but requests for terms not previously included are addressed to the Administrator/Author for possible subsequent entry.)
  - Student-initiated branching, bypass, and repeat actions.

The Student Course Workbook is coordinated to the display screen presentations. For each Session, it contains:
- Title page and specific reading assignments--Workbook, Student Text, PRM.
- Objectives, Topics (this session).
- Additional Reading Notes--text, figures, examples supplementing or amplifying the other reading sources.

COURSE DEVELOPMENT

Development of course presentation materials with the IIAS is facilitated by use of a set of standardized pre-printed worksheets. The Exhibits which follow the body of this paper illustrate five of the worksheet types which are more widely used in course development.

Exhibit A

The TEXT Worksheet is used to display textual matter to the student which in general does not require a response from him. Here, as in the other Worksheet types, the Author specifies the Session Number, Learning Type, and Topic Number for this display screen. This means that course material may be written out of the sequence in which it will appear to the student, and, in fact, may be developed concurrently by a team of co-authors. This information is not required on each sheet, when several belonging to the same topic are in physical sequence.

Note that the vertical and horizontal spacing of text in the display is controlled by the way the Author "fills in the blanks" of the Worksheet spaces.

Exhibit B

This example TRUE-FALSE Worksheet is used to state a Post-test question. Therefore, here as in the other question types, when used as a Pre-test or Post-test question,
- The HINT facility is not available to the student.
- There is no Wrong Answer analysis--the response is correct or it is not!
- There is no reinforcing reply or message to the student for a correct answer as there otherwise would be. The student is "rewarded" by omitting the tutorial or review/remedial matter.

Notice that there are certain operational options that may be specified by the Author:
- The use of certain control keys in

lieu of a T/F or Y/N typed response;

- The use of the "light pen" to avoid any use of keys or keyboard;
- Intensification, or highlighting, of any specified line of the screen;
- Acknowledgement of the student's reply with an audible tone ("alarm").

Only the shaded areas of the worksheets are required to have suitable entries.

Exhibit C

This QUESTION Worksheet is used for a completion question, in the Pre-test role. The question may be stated so as to require any free-form reply, consisting of a word, phrase, or numeric value.

As with the T-F example, full use of the QUES worksheet would require one or more expected Wrong Answer responses (with appropriate suggestive replies), as well as replies to the student for unanticipated answers and for HINT requests.

Exhibit D

This MATCH Worksheet is used as a "ranking" or sequencing Post-test question. The more general form of the Matching question requires the student to match items in two lists, such as synonyms, definitions, cause-effect, etc. Note that the number of items to be matched or ranked is variable, as specified by the Author.

Again, reinforcing or correcting messages may be provided when the match/rank criterion is met or is not met by the students' responses.

Exhibit E

The DEFN Worksheet is used to enter definitions into the on-line Glossary which is available to the student.

The "INSERT AFTER" box contains the instruction to IIAS to place this definition into the dictionary, after "I" in this instance. The first line contains the term or phrase being defined, "Interp&" in this example. Throughout all analyses of student responses by IIPS, such as in answers to questions and here in a request for definition, the '&' is a "don't care" character. That is, this definition, starting at Line 2 of the worksheet, will be displayed in response to a Glossary request for the root-word Interp, regardless of the suffix. The student may be asking for the definition of interpolate, interpolation, interpolating, etc. Other response-editing actions are either automatic or as explicitly specified by the author.

Exhibit F

Another TEXT Worksheet. Since this is a review display which the student will see only if he has missed the prior Post-test question, it is appropriate to redirect him to the reading assignment which he completed before starting the terminal work for this session.

Exhibit G

Another TEXT Worksheet, illustrating the manner in which the student is offered the option of bypassing the entire current session, yet retaining the ability to review one or more of the topics comprising this session. Of course, such a "fast path" offer accelerates the student's progress through the course.

Exhibit H

Each Session has a Summary screen such as defined by this TEXT Worksheet, containing a brief refresher for each of the topics covered. As the last statement indicates, this screen then leads to the offer to review selectively, at the student's discretion.

OTHER AUTHORING/PRESENTATION FACILITIES

The examples included here do not show the use of several other significant facilities available to the course author.

- One or more formal quizzes of up to 50 questions each may be included as intermediate or final exams. Review text may be displayed for wrong answers. Author-written remarks are presented for passing and for failing grades, as specified in constructing the quiz questions.
- Dynamic monitoring of course usage and student status is available to the author, by requesting reports which include:
  - Course utilization and student progress: number of current students, present location in course, number of course completions.
  - Course validation and item-analysis data: tallies of student responses to each question/ match/T-F, isolated by correct, wrong and unanticipated answers, to help in identifying improperly-worded or ambiguous questions.
  - Course completion data which relates average course time per student to the stand-up instruction that the present CAI course displaces.
  - Administrator/Author "electronic mail" responses to student-initiated comments.
  - Standardized system messages to the student instead of or in addition to those specified in the individual worksheets.

SUMMARY

This paper has presented the principles and concepts of course design and course authoring implemented in the current generation of computer software tools. Those activities do not require awareness of the computer's operations, since:

- The protocol of course logic is determined by selecting options appropriate to one's pedagogical strategies with X's or similar codes on pre-printed forms. Thereafter, it is controlled dynamically by IIPS, with no detailed action by the author.

- Content and presentation of educational materials are designed and entered from pre-formatted worksheets.

To sum up, computer-based instruction has found its place in instructional technology, and is here to stay.

REFERENCES

1. PLANCODE/I General Information Manual, Form GH19-2103
2. PLANCODE/I Program Reference Manual, Form SH19-1123
3. PLANCODE/I Student Text, Form SC20-1884
4. IIAS/IIPS General Information Manual, Form GH20-2446
5. IIAS Course Authoring Guide, Form SH20-2450
6. IIAS Simulation/Exercise Facility: Authoring Guide, Form SH20-2451
7. Interactive Instructional System Slide Set, Form GV20-0775

Acronyms in the above list are stated in full at Page Two of this paper.

Form numbers in the above list may be used to order the publications from the local office of IBM Corporation.

Ex 4, 6, 15 A - B

IBM

INTERACTIVE INSTRUCTIONAL SYSTEM
Text Display Worksheet

Page ___ of ___
Topic ___

INSERT AFTER 1 3 P 0 2

Session Number — Topic or Question Number

MC TEXT

Text

Review text for topic 2:

The "Expert Entry" facility is provided in PLANCODE to multiply the user's productivity. Why wait for the next question when you know what it will be? One's I.Q. does have a limit -- the Impatience Quotient, that is.

If you will be using PLANCODE at a location remote from your central computer site, "Expert Entry" may reduce your communication line costs as well.

The individual responses are separated by the # symbol.

T25 ☐ Branch Label

T26 ☐ Insert new label in front of text

IBM

INTERACTIVE INSTRUCTIONAL SYSTEM
True-False Question Worksheet

Page ___ of ___
Topic ___

INSERT AFTER 1 2 M 0 2

Session Number — Topic or Question Number

MC TF

Text of Question

TRUE - FALSE

The Modify, Insert and Delete commands are available for editing report statements in the Create as well as the Edit functions of Report Mode.

Answer is TRUE: A1 ☒ A2 ☐ PF Key (01-12) is TRUE — A3 ☐ PA Key (1-3) is TRUE

Answer is FALSE: A4 ☐ A5 ☐ PF Key (01-12) is FALSE — A6 ☐ PA Key (1-3) is FALSE

A7 ☐ Present detectable "TRUE" and "FALSE" above question text

Correct Answer: C1 ☐ Intensify line (Q1-Q5 or ANS) — C2 ☐ Alarm; C3 ☐ Common Message (001-999) — C4 ☐ Branch Label

Reply to Correct Answer: C5, C6, C7

Incorrect Answer: X1 ☐ Intensify line (Q1-Q5 or ANS) — X2 ☐ Alarm; X3 ☐ Common Message (001-999) — X4 ☐ Branch Label

Exit Message for Incorrect Answer: X5, X6, X7

Reply to HINT Request: H1, H2, H3

GEN

☐ = Required Entry

BEST COPY AVAILABLE

EXHIBIT C

Question Worksheet

INSERT AFTER 0 3 C 0 1
Section Number — Topic or Question Number

A = Introductory
C = Prepare
E = Teach
G = Quiz
M = Posttest
P or T = Review
W = Summary

GX20-1989-0
Printed in U.S.A.

Page ___ of ___
Topic ___

MC QUES

Text of Question

Q1 Topic 1. Modeling definition and example.
Q3 Please select the item which completes the following correctly:
Q6 PLANCODE/I model statements define calculations to be performed on data items.
Q7 These items (or variables) are identified in this modeling language by:
Q8 A. Programmer's language names.
Q9 B. User-selected names.
Q10 C. Number codes.
Q11 D. Code names.
Q12 Please key the letter for your completion choice.

Correct Answer

C1 B

C5 Number of words to match (1 9)
C7 Line detected is correct (Q1 Q12)
C6 PF Key (01 12) is correct
C8 PA Key (1 3) is correct
C9 Q12 Intensify line (Q1 Q12 or ANS)
C11 Common Message (001 999)
C10 Alarm
C12 Branch Label
C13 Send student to Exit

Reply to Correct Answer

Exit Options

X1 Intensify line (Q1 Q12 or ANS)
X3 Common Message (001 999)
X2 Alarm
X4 Branch Label

Text of Exit Message

Reply to first HINT Request

Reply to second HINT Request

IBM

Question Worksheet (cont)

Wrong Answer I

W2 Number of words to match (1 9)
W4 Line detected will match (Q1 Q12)
W3 PF Key (01 12) will match
W5 PA Key (1 3) will match
W6 Intensify line (Q1 Q12 or ANS)
W8 Common Message (001 999)
W7 Alarm
W9 Branch Label
W10 Send student to Exit

Reply to Wrong Answer I

Wrong Answer II

W15 Number of words to match (1 9)
W17 Line detected will match (Q1 Q12)
W16 PF Key (01 12) will match
W18 PA Key (1 3) will match
W19 Intensify line (Q1 Q12 or ANS)
W21 Common Message (001 999)
W20 Alarm
W22 Branch Label
W23 Send student to Exit

Reply to Wrong Answer II

Wrong Answer III

W28 Number of words to match (1 9)
W30 Line detected will match (Q1 Q12)
W29 PF Key (01 12) will match
W31 PA Key (1 3) will match
W32 Intensify line (Q1 Q12 or ANS)
W34 Common Message (001 999)
W33 Alarm
W35 Branch Label
W36 Send student to Exit

Reply to Wrong Answer III

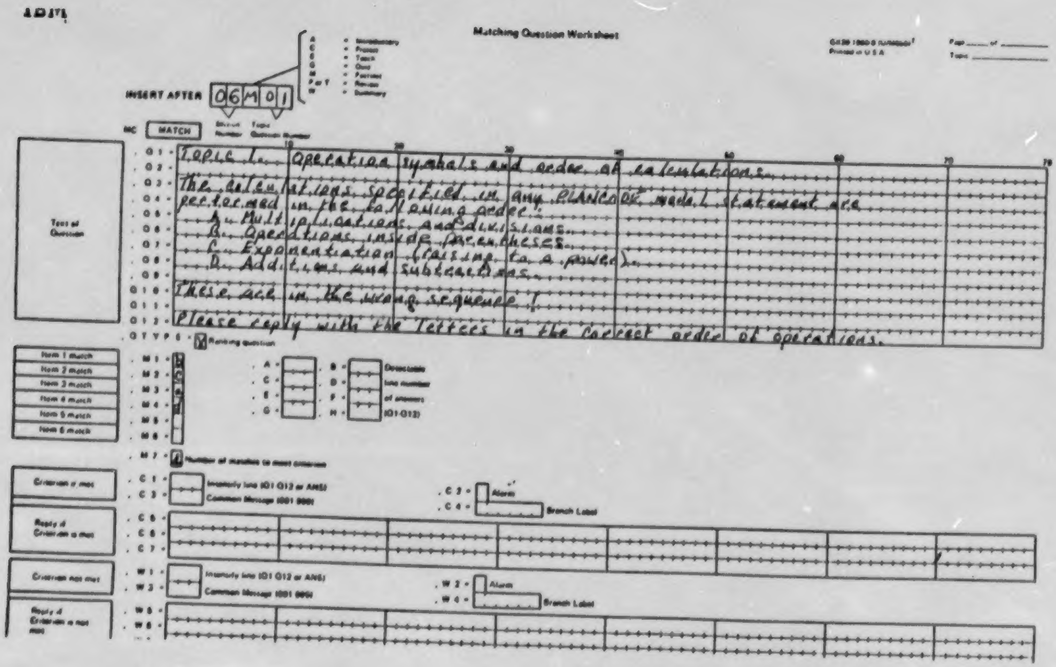
Wrong Answer IV

W41 Number of words to match (1 9)
W43 Line detected will match (Q1 Q12)
W42 PF Key (01 12) will match
W44 PA Key (1 3) will match
W45 Intensify line (Q1 Q12 or ANS)
W47 Common Message (001 999)
W46 Alarm
W48 Branch Label
W49 Send student to Exit

Reply to Wrong Answer IV

BEST COPY AVAILAB

EXHIBITS D-E



IBM

Matching Question Worksheet

Page ___ of ___

Topic ___

INSERT AFTER 0 6 M 0 1

MC MATCH

Q1 Topic 1. Operation symbols and order of calculations.

Q3 The calculations specified in any PLANCODE model statement are

Q4 performed in the following order:

Q5 A. Multiplications and divisions.

Q6 B. Operations inside parentheses.

Q7 C. Exponentiation (raising to a power).

Q8 D. Additions and subtractions.

Q10 These are in the wrong sequence!

Q12 Please reply with the letters in the correct order of operations.

QTYPE ☒ Ranking question

Item 1 match — M1 [illegible]

Item 2 match — M2 [illegible]

Item 3 match — M3 [illegible]

Item 4 match — M4 [illegible]

Item 5 match — M5

Item 6 match — M6

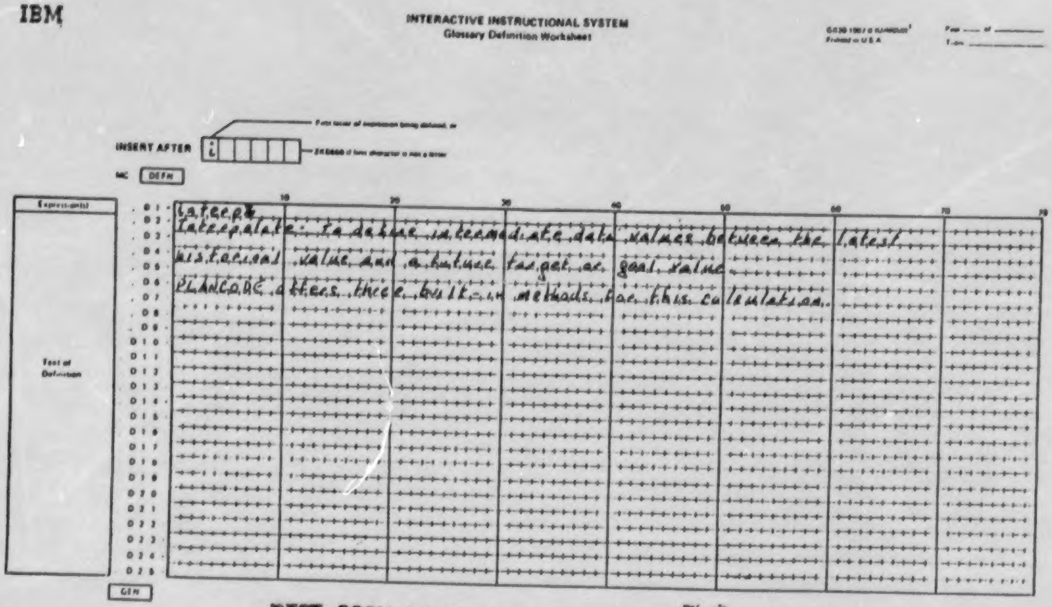
M7 4 Number of matches to meet criterion

Criterion is met — C1 Inventory line (Q1-Q12 or ANS) — C3 Common Message (001-999) — C2 Alarm — C4 Branch Label

Reply if Criterion is met — C5 C6 C7

Criterion not met — W1 Inventory line (Q1-Q12 or ANS) — W3 Common Message (001-999) — W2 Alarm — W4 Branch Label

Reply if Criterion is not met — W5 W6



IBM

INTERACTIVE INSTRUCTIONAL SYSTEM

Glossary Definition Worksheet

Page ___ of ___

Topic ___

INSERT AFTER i

MC DEFN

Expression(s)

Text of Definition

D1 interp

D2 interpolate: to define intermediate data values between the latest

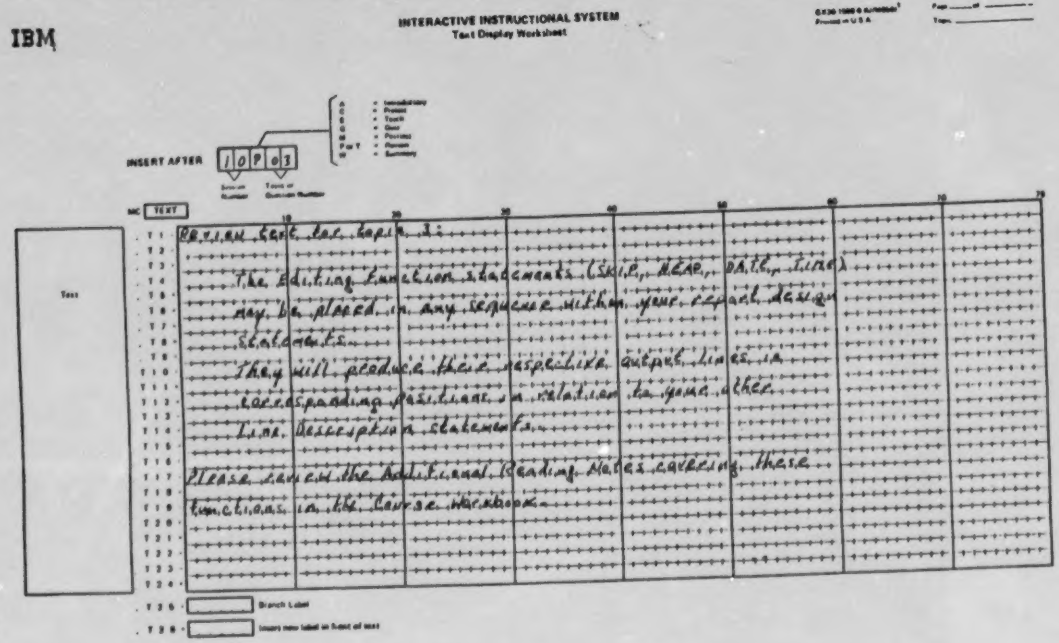
D4 historical value and a future target or goal value.

D6 PLANCODE offers three built-in methods for this calculation.

GEN

BEST COPY AVAILABLE

EXHIBITS F-G



IBM

INTERACTIVE INSTRUCTIONAL SYSTEM
Text Display Worksheet

Page ___ of ___
Topic ___

Introductory
Pretest
Teach
Quiz
Posttest
Review
Summary

INSERT AFTER 1 0 P 0 3
Session Number
Topic or Question Number

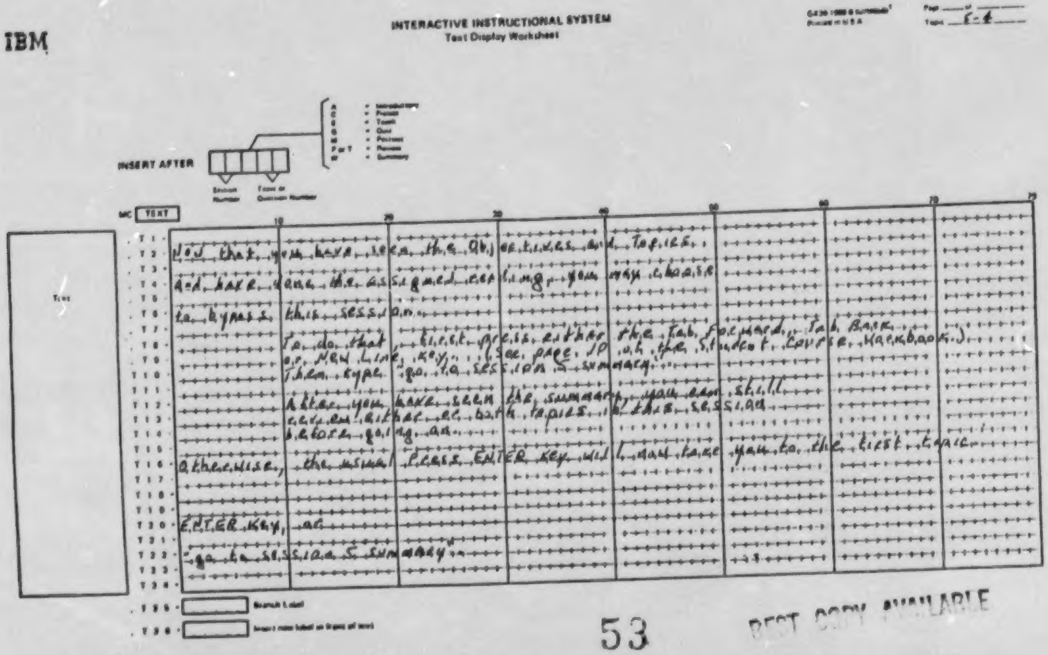
MC TEXT

Review text for topic 3:

The Editing function statements (SKIP, HEAD, DATE, TIME)
may be placed in any sequence within your report design
statements.
They will produce their respective output lines in
corresponding positions in relation to your other
Line Description statements.

Please review the Additional Reading Notes covering these
functions in the Course Workbook.

Branch Label
Insert new label in front of text



IBM

INTERACTIVE INSTRUCTIONAL SYSTEM
Text Display Worksheet

Page ___ of ___
Topic 5-4

Introductory
Pretest
Teach
Quiz
Posttest
Review
Summary

INSERT AFTER
Session Number
Topic or Question Number

MC TEXT

Now that you have seen the Objectives and Topics
and have done the assigned reading, you may choose
to bypass this session.
To do that, first press either the Tab Forward, Tab Back
or New Line key. (See page 10 of the Student Course Workbook.)
Then type "go to session 5 summary".
After you have seen the summary, you can still
review either or both topics in this session
before going on.
Otherwise, the usual press ENTER key will now take you to the first topic.

ENTER key, or
"go to session 5 summary".

Branch Label
Insert new label in front of text

BEST COPY AVAILABLE

IBM

INTERACTIVE INSTRUCTIONAL SYSTEM
Text Display Worksheet

Printed in U.S.A.
Page ___ of ___
Topic ___

INSERT AFTER 0 6 W 0 0

Session Number
Text or Question Number

- Introductory
- Pretest
- Text
- Quiz
- Posttest
- Review
- Summary

TEXT

In Session 6, we covered four main topics:

1. Operation symbols and order of calculations.
The PLANCODE calculation language has extensions
of the basic rules of Algebra, making it easy
to learn.

2. Data types and names.
User-chosen names for the data items appearing in the
model make statements more meaningful than codes would be.

3. Column addressing.
To provide flexible control of calculations, individual data
items are accessible by a consistent method of modifying row
or subset names.

4. The Data transfer Symbol.
The $ allows selection of model-specified lag or lead factors
in data calculations.

The next screen will ask you whether you want to review any of these
topics before going on.

Branch Label

Insert new label in front of text

EXHIBIT H

BEST COPY AVAILABLE

Special Session

ACADEMIC COMPUTING IN SMALL COLLEGES: PROBLEMS AND SOME SOLUTIONS

Sponsored by: ASCUE
Chaired by: Sister Mary K. Keller
Director, Computer Center Clarke College

ABSTRACT

ASCUE, as an organization, has had 15 years of experience in dealing with the problems of academic computing in the environment of a small college. The members, originally banded together to create a forum for intercollegiate help and exchange of expertise, have continued this sharing from the early days of limited resources to more recent improvement in hardware and software facilities.

Problems still remain. The panelists will take a look at some of these and share their experiences in solutions. Topics addressed will range over such matters as budget, space, personnel, setting priorities, use of packages, software development and documentation, and general academic support. The audience will be encouraged to raise questions about their own concerns relative to computing in similar environments.

Participants

F. Dudley Bryant, Associate Professor
Western Kentucky University

Jan Carver, Director Computer Center
Chatham College

Tom Moberg, Director Academic Computing
Grinnell College

Robert A. Rennert, Director Academic Computing
Kenyon College

Special Session

IMPACT OF COPYRIGHT LAWS ON SOFTWARE USE IN EDUCATION

Sponsored by: ACM SIGCAS & ACM SIGCUE
Chaired by: Ronald Anderson
University of Minnesota

ABSTRACT

This session will examine the impact of the copyright law in the educational use of microcomputer software. The views of private publishers and educational users will be used to explain why the law has developed into its current state. As for the do's and dont's, we will consider the basic problem of educators using microcomputer software: When is it legal to copy software? Do you need to purchase software for each hardware unit? Is it permissible to "network" several hardware units? Can you adapt software for specialized needs? Can you sell software, and if so, can you sell copies that you made yourself? How does leasing software differ from purchasing it with respect to your right to duplicate, adapt, and dispose of it? Finally we will point to other areas of copyright where special guidelines have been adopted for educational users and will suggest how similar guidelines might be appropriate with respect to microcomputer software so as to clear up the present confusion in the law.

The copyright laws and court interpretation seem pretty clear as to the legal issue of software copyright. Each user is entitled to the purchased software plus an archival copy to be used in the event the original disk fails.

There are three real issues that result from these laws:

1. Inconsistent implementation of the "archival copy" or backup copy among software publishers have thrown the situation into chaos. Some companies even avoid the entire copyright issue by not selling their software, but by "licensing" the software to users without backup copies.
2. Schools with multiple computers and the need for multiple copies of software are finding that software is "priced beyond their reach" and are not using their computers in the way they had intended. Schools with local computer networks are further confused as much of the available software cannot be used on their resource sharing systems.
3. A moral issue is raised when a law say a backup copy is allowed, a publisher denies such a copy, and a $40 program is avai able that allows the user to make a quasi-legal backup copy.

These issues and others need to be resolved to that publishers will earn their just profits, schools will have good software to use on their machines, and the children of the world can join the educational computer age with a smile on their faces.

Participants

LeRoy Finkel
San Mateo County, California
Office of Education

Thomas M. Gould
Cincinnati, Ohio
Gould, Reichert, Strauss & Cissel

Marjorie Kosel
Sunburst Publishing

Tutorial

COMPUTER LITERACY FOR THE CLASSROOM TEACHER GRADES K-8

Chaired by: Rachelle S. Heller,
Diane Martin
University of Maryland

ABSTRACT

This tutorial is designed to familarize classroom teachers with computer literacy topics and to help them become confident in introducing those topics into their own teaching environments. Topics will be offered which have been most requested by teachers. In addition, numerous field test activities for the classroom will be presented.

The tutorial will cover parts of the computer, history of computing, computer applications and integration of computer related activities into classroom curriculum. It will also emphasize problem-solving techniques and preprogramming activities.

Project Presentation

BASIC SKILLS IMPROVEMENT

Chaired by: Richard Pogue
Medical College of Georgia

Howard P. Alivar
Martha Cooper
Sandra K. Prigger
Michael G. Southwell
Glenn Yoshida

ABSTRACT

Using CAI to Teach Adults Basic Skills as a Prelude to Vocational Job Preparation Courses

Howard P. Alvir, Neil W. Carr, New York State Education Dept., 1600 Twin Tower, Albany, NY 12234

To identify the basic skills needed by adults who are unable to function in vocational education due to the adults illiteracy.

We kept scores by concentrating on gain scores rather than on total number of completers. This forced us to stress adult growth rather than entrance criteria which took only the most advanced of the illiterate adult population.

We used low cost microprocessors rather than high priced systems such as PLATO & TICCIT. We stressed self-standing systems that were user friendly and monitored by teachers who chould correlate the CAI with regular lesson plans. We wanted to set up a teacher/machine cooperation rather than stress teacherless lessons.

We achieved success in math gains; english gains, attendance gains; attitude gains; but not as much attitude gains as we had hoped. We feel that what we need is a more reliable instrument to measure attitude gains in a simple to understand scale.

ABSTRACT

Keybording (Touch Typewriting)

Martha Cooper, Coordinator, Special Programs, Northwest Missouri State University, Maryville, Missouri 64468
Mary Jane Sunkel, Chairperson, Business Education/Office Administration

At NWMS University, we are touch typewriting (keyboarding) using VT-100 display connected to a time sharing DEC PDP 11/70 computer. Although primarily designed for use in business education and office administration typing classrooms, the program could be used for teaching touch keyboarding to users in any discipline--a need existing because of the widespread use of computer terminals for entering data at the source. Students not only learn a touch system on the keyboard but also learn to work interactively with a computer, thus better preparing them for the job market.

During the fall of 1980 and 1981, we taught beginning typing classes with the computer. Students met in the computer terminal lab for the first part of the semester concentrating on basic skill-building. Then, they transferred to electric typewriters (IBM Selectrics) in the typing classroom for the remainder of the semester; this part of the semester included typing styles and followed more traditional methods. Skills carried over from terminal to typewriter; however, students seemed to prefer the use of the terminals.

Students key exercises, and the computer instantly compares the work keyed with a stored master file and indicates mistakes. Four levels (modes) of practice are available with each mode meeting a particular skill-building or instructional need. Two additional options are available for students working on speed and accuracy. A summary following each exercise indicates the number of words typed, the number of errors, percentage accuracy, and time required to complete the exercise.

All students work on the same computer account. Identification codes are assigned to each student, and results are stored in a packed integer data base file. The student can see his own summaries and tallies, but only the teacher has access to the results for all the students. Protection factors allow only the teacher to access the teacher's program for changing the context of the lessons, setting parameters for student performance, etc.

The keyboarding class is conducted in much the same way as the regular typewriting class except the teacher is relieved of such menial details of grading papers and recording scores. The classroom atmosphere is retained, the teacher's personality and competence are demonstrated, but more time is available to individuals problems. The computer is doing those things which computers do best checking for accuracy, counting errors, calculating percentages, comparing performance data to minimum standards set by the teacher, displaying results, and storing results.

Although the text of lessons can be easily changed, Northwest has used 31 short, sequential lessons. A 24-option menu in a special program for the teacher permits the changing of lessons, entering instructions to the student in a daily message, setting standards and requirements, and printing results.

The program also has been used on an indivudual basis in advanced typewriting courses for remedial practice by students who need to improve their keyboarding techniques, accuracy, and speed.

ABSTRACT

Computer Assisted Instruction in LD Math

Sandra Prigger, Marycrest College Department of Computer Science, Davenport, Iowa 52804

Taneyville R-2 School received a Title IVB Grant enabling the school to use the TRS 80 Model I, level II computer to implement computer assisted instruction in the Learning Disabled Classroom. At the start of the program each student was two or three grade levels behind in their basic math skills.

Each student attended the class for five hours a week. This time was divided into learning segments using the computer with CAI materials and learning games; as well as, regular classroom assignments and activities at their desks.

Each student was tested to verify what level or specific skills were needed. The students work on these skills until they are passed by use of the appropriate teacher made tests then the student advances to the next skill on the list.

The goal of the program is to increase the math level of the LD students using the computer and the described program by one and one half years for each year in the program.

ABSTRACT

CAI for Developmental Writing: Going Beyond the Fill-in Exercise

Michael G. Southwell, Department of English, York College/CUNY

Basic or developmental writers are students of high school or college age, or even older, with severe problems of clarity and correctness, usually arising from some sort of nonstandard speech background. Many teachers have thought of using computers to provide the enormous amounts of grammatical practice such students need. Most existing CAI programs, however, rely on simply asking students to fill in blanks, match answers, or choose a correct answer in a multiple choice format. This kind of instruction will never help developmental students to attain the productive understanding they need. But foreign language teaching techniques of supported composition can. These ask students to rewrite entire sentences and paragraphs, making various specified changes; in this way, students are acclimatized to create forms and meanings at the same time.

But how can such exercises be adapted to a CAI format? In general, the problem is to find a way to provide a stimulus or model, accept students' input, evaluate it, and then react appropriately. The obvious way to proceed would seem to be to program the computer to show the students a paragraph, let them type in their rewritten version, and then respond to it. But there are serious problems in using a computer this way. Developmental students usually can't even type at all, so typing even a simple sentence can be an agonizing and frustrating experience, highly suseptible to accidental errors. Asking such students to type a whole paragraph thus puts a tremendous burden on them. Further, the possibility of students' having omitted, run together, or erroneously separated words makes programming the computer even just to count words an extremely difficult task. Finally, evaluation of such whole text input is beyond the processing capabilities of many micro-computers in common use, which often do not have the necessary indexing and substring functions.

What alternatives are there? (1) The computer could provide the stimulus/model by phrases or sentences, and accept input the same way. But the point of the exercise is to help students deal with grammatical forms in the midst of a paragraph context. (2) A text editor could be made available, to let students fix mistakes in their answers before submitting them for evaluation by the computer. But such a system requires fairly complex hardware, and learning to use an editor only adds to the already heavy demands on the students' attention. (3) Students might work out answers on paper before typing them into the computer. But this would not remove the sheer physical difficulty of the typing, or the possibility of accidental mistakes, and would in addition tie up the machine for lengthy periods doing nothing but waiting for input. None of these solutions seems satisfactory.

However, it is possible to give students these kinds of contextual exercises in a CAI format if the computer is used to evaluate target answers only. Students can get the stimulus/model on paper. Then, if the computer is programmed to provide the interstices to the targeted words, students will need to type, and the computer will need to evaluate, only isolated words. This is a powerful way to use a computer to foster basic writing students' productive control of grammatical forms in the context of their own writing.

ABSTRACT

CAI and Polaroid Interactive Lecture System

Glenn Yoshida, Project Director, Associate Professor of Physiology, Los Angeles Southwest College, 1600 W. Imperial Highway, Los Angeles, CA 90047

Los Angeles Southwest College is implementing a two year project designed for students to correct deficiencies in their science background. Project activities include the development of learning modules using computer assisted instruction (CAI) and Polaroid Interactive Lecture System. Both instructional modes are self-paced, individualized, and provide immediate feedback to students. A series of color slides along with the audiotapes have been developed and will be utilized in conjunction with the instructional material on the computer screen. Faculty-developed modules include "Pollination and Fertilization", "Variation and Selection", "Plant Communities of S. Calif.", "Plant Ecology", and "Biological Terminology". A wide variety of lecture tapes have been prepared on the PILS. With this system, the student is able to run an audiotape which plays back the instructors voice while the unit concurrently write an almost exact replication of the instructor's notes or diagrams on a paper format. The faculty will explore, implement, and evaluate these techniques in order to improve the quality of preparation of students for graduation and careers in science.

Computer Based Education

INTERACTION IN LEARNING

Alfred Bork
Educational Technology Center
University of California
Irvine, California 92717
(714) 833-6911

The notion of interactive learning is one that is receiving considerable prominence these days, particularly in the development of either computer-based learning material or intelligent videodisc learning material. The idea that learning can and should be an interactive process, one in which the learner is a full-time participant rather than a spectator, is an old one in education, going back at least as far as Socrates.

The learning environment today, for almost all students, is not an interactive environment. The vast majority of learning is now done in lectures or with books. While a few good students can make the use of these media interactive, this is rare; for almost all students these methods are passive, a spectator activity rather than a participant activity.

No single view of the nature of learning has general acceptance. On the contrary, very divergent views are vigorously supported. But most learning theories suggest that it is better for students to play active roles in the learning process. In spite of this it has been difficult, given the number of students to educate, to provide such an active learning environment. For example, in the typical sixth grade class, likely to be a group of about twenty-five students, it is clear that each student cannot receive much individual attention from the teacher. The possibility of each student learning in a Socratic fashion, through interaction with the teacher, is simply out of the question, given typical class size and typical teachers. Rather, passive learning modes dominate. The situation is even worse in the large classes which are necessary in most universities. Indeed, when class size gets above about ten, few teachers can maintain an interactive environment. So even in schools that pride themselves on being small, the learning situation is seldom interactive.

COMPUTERS

Modern technology applied to the learning process, particularly the use of computers, either alone or in combination with optical videodiscs, allows the possibility of an interactive learning environment for all students. The developer of materials in this area is often thrilled with these interactive possibilities, since no interaction has been possible in the use of many of the older media in which they worked. An unfortunate tendency exists to view any interaction as marvelous! The major purpose of this paper is to try to point out that there are levels at which interactions can occur, various degrees of interaction, and that it is likely that the levels of greater interaction are ones which will have the more profound effects eventually in the creation of learning materials. Not all interactions are equal!

FACTORS DETERMINING INTERACTION LEVEL

Two aspects concerning interaction in computer-based learning material will be considered in this paper. First, we will look at the quality of each interaction. Three factors are considered here; the type of input required of the student during the interaction, the method of analysis of this input, and the action taken in the program after the input. The second aspect might be termed the overall level of interaction.

Although we will discuss a possible increase of interaction, this measure is by itself not the focus of this paper.

Rather, we want to show developers the range of possibilities in the hope that they will move to more sophisticated levels of interaction.

USER INPUT FORMS

We can distinguish a variety of ways in which a student can interact with a computer-based learning program. We will consider first the types of input which are possible. These types are listed roughly in order of increasing level of interaction, starting with very simple types of user interaction with the program and ending with very complicated types.

The program preceding such an interaction may have done a variety of things. It may have shown some material to the student and asked a question. It may have allowed the student to ask certain types of questions. In any case we would expect that there would be many such interactions within the program if the program is to be considered interactive; we will later discuss the number of interactions.

The following list shows types of user input forms which can be considered. As indicated, we have tried to rank them from less sophisticated interactions to more sophisticated interactions. It would be difficult to defend the exact ordering in this regard, but the overall ordering is, I believe, clear.

1) Multiple choice
2) Multiple choices
3) Yes-no?
4) Concealed multiple choice
5) Fill in the blank in a sentence
6) Pointing input to identify a feature
7) Numerical input
8) Natural language input

It is actually hard to distinguish between yes-no and multiple choice, since yes-no is a form a multiple choice. However, in spite of this, our experience is that yes-no in general is a much more pleasant form of interaction than is multiple choice. For reasons that will not be fully discussed, our group at Irvine almost never uses multiple choice in any of its programs, whether quizzes or any other type. We regard multiple choice as a particularly inhumane form of input, leading to guessing activities having nothing to do with purposes of learning.

The term "multiple choices" implies the possibility of having several right answers. The situation is slightly better than for multiple choice, but this is still a form which the Educational Technology Center considers to be a weak form of interaction. There is not quite as much guessing involved in the process, but still a considerable level. One need only investigate briefly the literature available for students about how to take tests (mostly meaning how to take tests with either multiple choice or multiple choices) to see how many factors are involved in taking such tests that are completely independent of knowing the material.

We have found the concealed multiple choice strategy to be very useful. This strategy was first developed many years ago by one of my colleagues, Stephen Franklin, in connection with on-line quizzes in the precalculus mathematics class. It is used extensively in our physics quizzes. The basic notion is that the computer has stored, or can generate, a large number of both right and wrong answers to a question. These are presented to the student in a random order. The student sees only one possible answer at a time and must make a decision about the correctness of each answer before seeing the next possible answer. The process can be stopped in at least two different ways. One is to give some fixed number of possible answers; the other is to continue until either the student correctly identifies a correct answer or makes a mistake. This form of input overcomes many of the difficulties of multiple choice. It is particularly useful in quizzes.

While we would not want to use the "fill in the blank" approach in all or most inputs, it is often a convenient strategy with students. It has the advantage that almost all students are familiar with it from workbooks and other precomputer materials. Furthermore, it is a "comfortable" form of input in some situations, with the student having a pretty good idea of what is demanded. When fill in the blank is used on the computer, it can be used in a very natural form, with an underlined blank at the right location and with the input cursor moved to this blank.

Numerical inputs are frequent in mathematical and scientific contexts. Often with a numerical input, one is asking for the solution to a problem.

The most sophisticated form of input listed is natural language input. Here, however, we must be cautious. Natural language input in this context means natural language as seen by the student. What techniques are used internally in the program to analyze, for example, an English sentence typed in by the user, are an entirely different issue. In most of our computer displays we have not found it necessary to parse English in the sense that is possible in modern artificial intelligence. Rather, simple string matching, with some logic, is often sufficient in identifying correctly what appears to the student to be a very free-form input. We will discuss this further later in the paper.

Pointing input may be simply another variant of some of the other forms, such as multiple choice. But it can also be used in ways that are difficult to use in verbal input at all. Thus, if we are trying to get a student to see a key feature of a graph, for example, a place where the graph has a maximum or a minimum, pointing input may be the best way to do this. The type of pointing input will vary with the computing system, but fundamentally from the student point of view, the input is the same whether touch input is used, a mouse, a joystick, paddle wheels, or other devices.

ANALYSIS OF USER INPUT

The next stage in determining how interactive a program is is to ask what analysis is conducted internally in the program each time the user inputs something. We have already touched slightly on some of these factors in the last section on forms of student input. The following possibilities for analysis might be considered. Again, we are to some extent listing them in terms of increasingly sophisticated interaction, although again the exact difference between neighboring items is vague.

1. None
2. Exact string match
3. Exact match of substring
4. Partial match
5. String matches with logic
6. Possible processing of string
7. Parsing of natural language

The listing of "no analysis" may be suprising to some readers. But with an actual computer-based learning program, this is sometimes a useful way to proceed. We may ask a question of a student with no intent of analyzing the student response. For example, early in a program, one might want students to simply spend some time thinking about a critical issue. One way to obtain such thinking is to ask the right question and then let the student type in a response. At this stage one may not try to analyze this response at all because it was the student process of attempting to answer it that was really the important thing, not the answer itself.

The second type of analysis is an exact string match; we look for a match with the student input and a predetermined string. This is seldom used in our programs, since it assumes that students will be extremely disciplined in exactly what they will enter to the computer. A more common strategy would be to match only a substring, but even this is a rather simple-minded matching technique. Usually many such matches will be necessary to each input.

Some projects have used partial matches; that is, they have a facility which says "accept this if it occurs in the student input with only one letter off or two letters off." The attempt in such a case is to overcome possible spelling or typing problems. We have found that we can obtain quite good success by a simpler procedure, never testing on whole words but only on characteristic pieces of words.

The most common matches used in much contemporary computer-based learning material are a series of string matches with some logic. That is, one asks "does the string contain this string and that string but not some third string?" Or one asks, "does it have this string followed by this string?" One can make quite sophisticated analyses of student input, fully adequate in most circumstances, by relatively simple use of such logic.

In all these cases the developers may want to precede the matching with some processing of the string. Thus if one is looking for both a number and some units attached to the number, the processing might consist of first separating off these two pieces and proceeding to analyze each independently. Or it may be desirable to remove certain material in the student input string, perhaps after it has been recognized, and work only with the rest of the string. Or it may be desirable to split a string at a certain symbol, word, or number and analyze the parts separately.

We have little new to say about analysis of numerical input, most common in programs in the sciences and mathematics. In checking such an input, the designers of the program should allow a reasonable possible range for the answer, as an exact match will often be unlikely.

We have already commented that in most computer based learning material full parsing, both syntactically and semantically, of natural language is usually not possible. In most cases in our work we have found it unnecessary. However, as our techniques in this direction improve and we understand more about how to build intelligence within even small programs, it may be that this type of interaction will become more important.

ACTION AFTER INPUT

We have a variety of possible actions that can be taken in the program after the input. Here the listing becomes somewhat more uncertain as to how to classify these, although some are more desirable than others. Here is the list.

1. None
2. Right-wrong reply
3. Right-wrong reply with additional information
4. Aid sequences
5. Linear action
6. Nonlinear action

Again, the possibility that there may

be no action after input, with or without analysis of the student response, may be strange to many. However if one considers conversations between human beings, one will note that this is often the case. A human being does not constantly say, in conversing with another human being, "right!" Rather, you may often simply go ahead to another question, with no immediate feedback. Sometimes the nature of the student input will be noted within the program, and so can affect either future statements by the program or even the entire flow of action in the program. The goal is a quality of interaction which resembles that between a highly trained, interactive teacher and one or a few students.

Undoubtedly the most common interaction found in computer-based learning material is a right-wrong response. In fact, one sees many programs in which in every response the student is told immediately whether the answer is right or wrong. Some schools of psychology may even favor such an approach.

But our group at Irvine does not believe that it is necessary. A right-wrong reply is often not very helpful and may, in fact, discourage the student in some circumstances. If a right-wrong reply is used frequently, we consider it desirable to change the vocabulary. For example, one can have a series of different "good" messages available in the program, perhaps at different levels of praise. This is particularly useful in areas of the program where much repetition occurs, because the code is used over and over. It also has the advantage that the developer of the material does not need to spend large amounts of time at each action after input deciding what type of good message to give to the student.

A right-wrong message will usually be of more help if it comes with additional information. Again the model of a human teacher should be followed. When we read the Socratic dialogs, Socrates's reply to a student answer to one of his questions was often long and often proceeded eventually to a different question.

The possibility of extensive aid to the student is one that certainly needs to be considered frequently in developing computer-based learning material. Sometimes this aid may be remedial in nature. Thus, in physics material, one may discover that a student lacks some critical mathematical background. It is possible to have a sequence at this point in the program that develops that mathematical background, and thus assure that the student can proceed with full understanding.

The distinction between linear and nonlinear approaches is an important one. In the linear approach, each student passes through the same material; that is, after a given question the next question will be the same for all students. In a highly nonlinear environment, the student inputs may alter the flow of the material and so different students may find themselves in different tracks, with different approaches. Undoubtedly some nonlinearity is desirable to take into account individual differences. But most existing computer-based learning material is not highly nonlinear, simply because we have little experience so far in creating such highly nonlinear material.

MEASURING THE NUMBER OF INTERACTIONS

So far we have considered only a single interaction and talked about a number of ways in which that interaction could be classified. Based on these classifications, some interactions will represent, therefore, a high degree of interaction and others a very low level of interaction. Thus, a typical low-level interaction would be a multiple choice question with only a right-wrong response.

However, a program does not contain a single student input. Rather, a full program has many different student inputs, and therefore many different interaction points. Hence, in developing a scale for measuring interaction, it is important to ask not only about the quality of each interaction, but to get some overall information about the number of interactions. A program which has ten minutes of no interaction with the student, and then has one extremely good interaction and then ten additional minutes of no interaction, is not an interactive program.

In developing a total measure for interaction then, we need to have some measure for each interaction or perhaps for an average interaction in a program and some measure of the number of interactions. It is the question of the number of interactions that we consider in this section.

There are at least two ways of measuring the number of interactions. One is to ask: what is the time between two interactions on the average for an average student? This number could actually be determined empirically, simply by observing students using the program. Undoubtedly existing material will differ greatly in this interaction time factor. This is probably the most reliable way of measuring the number of interaction within a program.

Another possibility, with some difficulties, is to ask how many interactions occur on a single screen. This mode of working was satisfactory in an environment where the whole screen had to be erased at a single time, such as with direct view storage tubes. But with more modern raster-based displays, often portions of a screen are erased, so there is not an entirely clear view of what constitutes "a screen." Hence, it seems more difficult to measure interactions in this fashion.

A TENTATIVE NUMBER OF INTERACTIONS

Based on these considerations we now can offer a tentative measure for the quality of interaction in a program. The factors involved have already been discussed, and I will not try to pin them down further numerically. The time between interactions must be taken into account reciprocally, since the smaller the time between interactions that occurs, the greater the interaction of the program. Hence, one possible measure is as follows:

$$\frac{\text{Quality of the Average Interaction}}{\text{Time Between Interactions on the Average}}$$

Perhaps the time should be given additional weight.

I stress that the exact numerical details are not as important as is the qualitative difference. We do not need to assign a measure to interaction correct to three or four decimal places! Rather, we need to understand that some programs are more interactive than others, and that interaction is in general a desirable feature for learning materials.

The author welcomes further discussion on this issue.

INTEGRATING CAI INTO THE CURRICULUM

Glyn Holmes and Marilyn Kidd

Language Laboratories
University of Western Ontario
London, Ont. N6A 3K7, Canada
(519) 679-2555

Until the advent of the microcomputer, relatively few educational institutions had incorporated computer-assisted instruction (CAI) into the curriculum. Though it has not solved many of the other problems of CAI, the microcomputer has at least offered a way around the previously prohibitive cost of acquiring hardware. As a result, some educational institutions have already invested in several microcomputers and have active CAI programs. Others are on the verge of making a decision but are unsure about the advisability of such a move. Educational technology, as we all know, has not always lived up to expectations. The discussion that follows is directed at these would be implementors. It is not a defense of, or an indictment against, CAI but, rather, an attempt to highlight some important questions that should be raised when contemplating the implementation of a CAI facility.

1. THE BASIC QUESTION: WHY IMPLEMENT?

An interest in the implementation of CAI may evolve from a variety of situations. Educators may consider it a possible solution to a particular problem existing within their institution, i.e. overcrowded classrooms, a need for remedial or enrichment facilities. On the other hand, although no specific need exists, they may find it desirable to acquire this new piece of instructional equipment in order to further enhance an already effective system. Whatever the reasons, CAI has attracted attention because it appears to offer many advantages to all those involved in the instructional process, not only students but also teachers.

Since its inception the computer has always held great potential for individualized learning. It has been stated that one hour per day with a computer can provide a student with more interaction--hence personal attention--than a day in the regular classroom (Atkinson, 1967). The ability to respond to student input and thereby engage in two-way interaction is, in fact, a feature unique to CAI as a form of mechanized instruction. Because the computer is able to analyze each student's response it can also react appropriately to a variety of student errors. Its large storage and retrieval capacity provide immediate accessibility to a wealth of useful data.

From the point of view of the teacher, the computer has the advantage of relieving him of many of the tedious drill-and-practice sessions that often devour classroom time. With the aid of this tireless and constantly available tutor the teacher can devote more time to the more creative aspects of teaching of which only human instructors are capable. The computer will also assist the teacher in keeping records on student performance and, in some instances, creating and marking tests.

The question as to the effectiveness of CAI in terms of student performance inevitably arises. Studies to date have not proven conclusive. Most evidence seems to indicate that CAI is at least as effective as traditional instruction, and some studies indicate that with CAI the rate of learning can be accelerated (Magison, 1978). There are innumerable reports of CAI improving student motivation and in some instances it has had success with students who performed badly when more traditional methods were used.

One of the difficulties involved in

any attempt to determine the effectiveness of CAI is that the results often fail to differentiate the various aspects concerned. The ultimate effectiveness of CAI is closely linked to the quality of the courseware materials (both pedagogical content and programming sophistication should be taken into account), the place of the computer in the existing curriculum, the quality and availability of the equipment used, and the transmission to the users of positive or negative attitudes towards CAI.

2. COURSEWARE

It has been acknowledged that the lack of adequate courseware is a significant contributing factor as to why CAI has not been as successful as was first anticipated (Blaschke, 1979). The problem of obtaining courseware is one that the implementor should confront even before the purchase of any equipment, lest he find himself in the embarrassing dilemma implied by the title of a recent article, "Computers in the Schools: Now That We Have Them...? (Eisele, 1981).

There are several sources of courseware, each offering its own advantages and drawbacks. One possibility is the purchase of readymade packages. This source has the advantage of allowing a CAI facility to become operational quite quickly and with a minimum of cost. A number of CAI marketing agencies are now beginning to make their appearance to fill the demand. In a recently published article by Constance Curtin (1981) a useful list of microcomputer publishers is indicated. The disadvantages of such readymade packages can be considerable. Usually they are not available to run on all makes of computer. Frequently the lessons are quite short or incomplete, and and are not suitable for use as an integral part of a course. As well, the lessons may be devised for use in conjunction only with a specific text, or their methodological bias may be unsuitable.

A second possibility that is becoming increasingly popular is to obtain an externally-produced template system; that is, a readymade program which allows the teacher/author to insert his own pedagogical materials into predetermined lessons formats, such as multiple choice, fill-in-the-blank, transformation, Cloze tests etc. It is possible to obtain a template system which is potentially applicable to all disciplines, or one which is designed for a specific subject area. The advantages are that the teacher can, in a relatively short space of time and with little knowledge of computer programming, tailor CAI materials to the needs of his own students. The price that must be paid is a certain standardization and therefore inflexibility in the lesson formats. Possible user boredom is also a factor that has to be taken into consideration.

The most satisfying type of courseware is undoubtedly that which is custom made to one's own specific needs. It is also the most costly in terms of the time, energy, and money required. In-house production need not, however, be a pipedream, even for the individual teacher. In the past exaggerated claims have been made about the size and the composition of the team required to produce quality CAI courseware (Watson, 1972; Hicks and Hunka, 1972). Experience is showing that it is not necessary to engage a bevy of specialists, just as it is being recognized that it is not necessary to exploit all the technical features of the computer in order to produce a good CAI lesson. There is strong evidence that effective, motivational learning environments can be created using simple techniques (Roblyer, 1978). Many innovative teachers are beginning to realize that they can be their own educational psychologist, media specialist, subject expert, and computer programmer. It is perhaps this kind of creative involvement in CAI that will best assure its success.

3. THE ROLE IN THE CURRICULUM

Some CAI projects are open to criticism with respect to the role that CAI has been given in the curriculum. On several occasions the computer's function has remained quite nebulous. The result has often been sporadic and ineffectual use by the students. Proponents of a systems approach to curriculum planning insist that elements of the teaching/learning system should be viewed as an entity consisting of interdependent parts (Banathy, 1969). It has been suggested that the computer needs to be likewise integrated for it to be truly effective (Carpenter, 1970). The role the computer is to assume should be decided upon before implementation as this role will determine, to a great extent, the type and quantity of equipment required and the nature of courseware design. In general, the computer can be used in one of two ways: a mainline use in which it provides the principal source of instruction or an adjunctive use in which the computer serves merely to supplement traditional classroom instruction.

Although various institutions have adopted one or other of these methods according to their particular requirements and both methods have met with varying degrees of success, no major studies had been conducted to determine the relative effectiveness of each method until the late 1970s. At this time an experiment was performed using the TICCIT system as an example of mainline instruction, and the PLATO system to illustrate adjunctive use (Alderman et alia, 1978). It was discovered that using the computer as a main source of instruction had a negative effect on student course completion rates, but those who did complete the course did better on the average than those who had received classroom instruction. The adjunctive use of the computer showed no consistent effects (positive or negative) on either completion or achievement. Some additional factors influencing these results were the large discrepancies in the quality of the courseware available and the attitude of various instructors, some of whom did not give students much positive encouragement to use the computer facility. It should also be noted that completion rates improved in the mainline use of the computer when teacher involvement was increased. We suspect, as well, that the results for the adjunctive use would have been more positive had the computer been completely integrated into the curriculum rather than being left to the whim of the individual. These results, although somewhat inconclusive as to the most effective method for use of the computer, nevertheless, quite definitely indicate the vital role of the teacher in any form of instruction.

4. THE CHOICE OF SYSTEM

Although the educational function the computer will serve should dictate the type of equipment to be purchased, in some school settings the implementor may find himself with no choice of hardware, for the local board may have an existing computer network and expect that any CAI project utilize the same machinery. This attempt to merge administrative and instructional interests seems to be prevalent, and in one survey the majority of respondents felt that the computer facilities would have to be multipurpose in order for the local school district to loosen its purse strings (Blaschke, 1979). Though dictated by financial stringency, systemsharing often results in the relegation of one application, usually instruction, to another. All too often the system is geared to administrative functions, and programming help is made available only for administrative purposes. The CAI innovator is left with a system for which there is little or no available courseware, and without the wherewithall to produce it.

In other instances the process of implementation may become bogged down in the problem of choice of hardware. The industry moves so quickly that any one machine seems obsolete almost as soon as it is put on the market. The concern with obsolescence, if not held in check, may stifle action. Such fears--illogical insofar as an obsolete machine will still continue to serve the purpose for which it was purchased--highlights a widespread preoccupation with the hardware system itself, divorced from its particular applications. As far as any envisioned CAI system is concerned, hardware preoccupations are misguided. The primary consideration in CAI should be that of the instructional purposes for which it will be used and the amount of courseware availabe for this system.

These priorities must be applied when, as is frequently the case, only one system can be purchased. Perhaps the ideal solution is to install a facility that includes more than one make of computer. A CAI facility that has, for example, two or three different microcomputers, each of which can provide desirable courseware, can deliver a significantly more useful service than one which is dedicated to a single type of machine. A multi-machine facility is certainly not beyond the means of many institutions, especially in view of the large sums that are being spent on computing.

5. COSTS

In spite of the wistful hope of some educators that the improvement of the instructional process be the prime consideration in instructional change (Kropp, 1970), cost--specifically, cost-effectiveness--is the primary goal. CAI must either offer the same educational product for less money, or offer a better product. At the same time it must be realized that the "product" does not have to be measurable student performance; the improvement might be in the educational environment. Many costly CAI projects have folded when special funding was exhausted, and they came under the scrutiny of the regular budgetary process.

The purchase of a CAI system has, of course, become a far less formidable proposition with the advent of the microcomputer, but hardware represents only one--albeit the most visible--component of a CAI system. The detailed, though now dated, cost estimates of Kopstein and Seidel (1969) are still useful in identifying some of the not-so-visible costs that can cripple a project: the replacement and maintenance of hardware, and the purchase or production of courseware. Maintenance involves not only repairs, and possibly shipping machinery to outside repair centers, but also the regular replacement of deteriorating tape or disk storage facilities.

Eventually the system itself must be replaced. Courseware costs can vary enormously according to the source, as we have discussed earlier.

The drawing up of cost estimates is of vital importance to any implementation scheme. Unfortunately the experience of one institution may not be of help to another, given all the variables. Estimates must be done on an institution to institution basis (Levien, 1972).

6. PEOPLE AND ATTITUDES

"The success of change is dependent solely on what people do and are prepared to do," write Fullan and Park (1981). Educators wishing to implement a CAI system should be fully aware of the critical importance of people's attitudes, and should not underestimate the ability of one's colleagues to resist change.

a) Teachers

Studies have indicated that the second greatest obstacle, after cost, to the implementation of CAI is likely to be teacher acceptance (Rockart and Scott Morton, 1975; Wilcox, cited in Levien, 1972). Their active co-operation is vital during the whole of the implementation phase, but unfortunately is not always easy to obtain.

One probable obstacle to their willing co-operation is the fact that the reward structure for teachers rarely motivates them to implement (Oettinger, 1969). A teacher is usually rewarded for years of satisfactory service rather than for the time, energy, and initiative required for implementation. On the other hand, Leithwood et alia (1979) have identified possible altruistic sources of satisfaction that can induce teachers to adopt change: if the change provides students with more of the same experiences traditionally associated with the classroom, if it increases student motivation, and if it increases student competence. As indicated above, there is evidence that CAI does produce these effects.

Many teachers are justifiably prey to more egocentric concerns of which the implementor needs take stock. The teacher, often inspired by ignorance, may have a negative perception of his changed role in any new scheme of things. Will a CAI facility reduce the importance and meaningfulness of his role? Of course the teacher's concerns may appear more or less legitimate according to the proposed role of CAI in the curriculum, but in any case the implementor would gain little by trying to conceal the fact that the computer can perform some of the functions expected of the teacher. Indeed, this is perhaps the only justification for any form of educational technology. At times of objectivity, most teachers would admit that some of their multiple duties are menial and repetitive. All have at one time or other wished to be freed of these tasks in order to concentrate on more creative activities such as live interaction, or the preparation of teaching/learning materials. To a large extent the book, chalk and blackboard, moving pictures, and the like, have contributed to this liberation process. The computer is simply the latest machine to offer its services.

Another fear not uncommon to teachers is that they will ultimately be replaced by teaching machines. The virtual demise of instructional television as conceived in the 1960s should provide clear evidence of the pivotal role of human interaction in education. It has been said, perhaps with justification, that a good machine cannot replace a bad teacher (Kropp, 1970), and that CAI can only support secondary educational objectives, namely the teaching of things (Thomsen, 1970).

An allied fear, especially prevalent in the humanities, is that encroaching technology is synonymous with progressive dehumanization. Student users on the contrary indicated in large numbers (88%) that they did not find this to be so (Alderman et alia, 1978). Fear of technology destroying humanity has of course been with us since time immemorial: Socrates is reputed to have believed that written documents would cause the decline of memory (Gerard, 1969). Yet man seems to have survived all technical advances--the book, the telephone, locomotion, television--and kept his humanity relatively intact.

b) Students

It might justifiably be argued that students are more naturally receptive to change than teachers; they are not responsible for the smooth operation of the institution, and they are generally not as prone to ingrained habits and irrational fears as are teachers. However, they do have basic educational requirements which the CAI facility must be geared to meet.

Perhaps the most vital learner need is for quality learning materials (Holmes and Kidd, 1981). If the materials being used are not perceived as helping him achieve his learning objectives, the learner will reject them. They must be pertinent to his course of study, preferably addressing core, rather than peripheral issues. They must also provide a learning experience that cannot be duplicated by other more practical, or more convenient, means. Experience has shown, for example, that students of French respond more positively to CAI drills on grammatical points included on their final examination, than to vocabulary drills using items on which they will not be specifically tested. Perhaps of necessity, students' concerns are pragmatic.

We have tried to emphasize elsewhere (Holmes and Kidd, 1981) the importance of providing for learner convenience in a CAI facility. It may be all too easy to overlook the fact that learners will exhibit decidedly negative reactions to a facility if they do not have easy access to it, if they are delayed because of insufficient machines to meet demand, if the machinery is unreliable, if help is not readily at hand, if it is difficult to access the CAI program, if the equipment and pedagogical materials do not allow for a large measure of user control. In many respects these learner needs, though understandable, can place a severe strain on a facility, particularly if they have not been foreseen in the planning stage.

c) Administrators

Finally, a positive stance by administrators is necessary before any implementation can get off the ground, for they hold the purse strings. Teachers tend to look to them for guidance in priorities and policymaking. Factors which may be instrumental in their approval of CAI implementation have been identified by Leithwood et alia (1979) such as bureaucratic safety, peer approval, and attraction of outside resources.

Basically, administrators will proceed with caution; their mandate is to oversee the operational continuity of the existing system, and they will tend to reject anything that they see as potentially disruptive (Margolin and Misch, 1970). Unfortunately change is always potentially disruptive, and any implementation must offer other advantages to the administrator. As has been suggested, peer approval and outside funding are possible rewards for the administrator. Both have the added advantage of offering him means of externalizing the responsibility for implementation. Outside funding implies that someone else has first appraised the project and deemed it worthy, and peer approval provides the kind of moral support that may entice one to make a move. If the implementor can therefore find some outside funding from, say, a charitable foundation or a government agency, and propose the type of facility already in existence at a prestigious (or perhaps even a competing) institution, his chances of success may well be improved.

7. CONCLUSION

These, then, are the basic issues to be considered when contemplating the implementation of a CAI system. Due to the constraints of time and space this discussion cannot hope to be exhaustive, neither can it hope to be totally objective for our own involvement in CAI indicates a definite conviction, on our part, of its effectiveness. We do hope, however, to have helped crystallize the questions any prospective implementor should be asking himself, and as well to have alerted him to some of the problems he can expect to encounter.

REFERENCES

Alderman, D.L., L.R. Appel, R.T. Murphy. "PLATO and TICCIT: An Evaluation of CAI in the Community College." Educational Technology, 18, 4 (1978), 40-49.

Atkinson, R.C. "CAI-Learning Aspects." Computers and Education. Ed. R.W. Gerard. New York: McGraw Hill, 1967, pp. 11-63.

Banathy, B.H. Instructional Systems. Fearon: Palo Alto, CA, 1968.

Blaschke, C.L. "Microcomputer Software Development for Schools: What, Who, How?" Educational Technology, 19, 10 (1979), 26-28.

Carpenter, C.R. "Computer-Regulated Learning." Computers in the Classroom. Eds. J.B. Margolin and M.R. Misch. New York: Spartan Books, 1970, pp.163-75.

Curtin, C. "Publication Practices for Microcomputer Programs." Journal of Computer-Based Instruction, 7, 4 (1981), 123-29.

Eisele, J. "Computers in the Schools: Now That We Have Them ...?" Educational Technology, 21, 10 (1981), 24-28.

Fullan, M., P. Park. Curriculum Implementation: A Resource Booklet. Toronto: Ministry of Education, 1981.

Gerard, R.W. "Shaping the Mind: Computers in Education." Computer-Assisted Instruction. Eds. R.C. Atkinson and H.A. Wilson. New York: Academic Press, 1969, pp. 15-39.

Hicks, B.L., S. Hunka. The Teacher and the Computer. Philadelphia: W.B. Saunders, 1972.

Holmes, G., M.E. Kidd. "Serving Learner Needs: From Teletype to Micro." System, 9,2 (1981), 125-32. Kopstein, F., R.J. Seidel. Computer-Administered Instruction Versus Traditionally Administered Instruction: Economics." Computer-Assisted Instruction. Eds. R.C. Atkinson and H.A. Wilson. New York: Academic Press, 1969, pp. 327-59.

Kropp, R.P. "Making CAI Work." Computers in the Classroom. Eds. J.B. Margolin and M.R. Misch. New York: Spartan Books, 1970, pp. 205-22.

Leithwood, K.A., M. Holmes, D.J. Montgomery. Helping Schools Change Derived From Field Experience. Toronto: OISE, 1979.

Levien, R.E. The Emerging Technology: Instructional Uses of the Computer in Higher Education. New York: McGraw Hill, 1972.

Magison. E.M. "Trends in Computer-Assisted Education." Educational Technology, 18, 4 (1978), 5-8.

Margolin, J.B., M.R. Misch (eds.). Computers in the Classroom. New York: Spartan Books, 1970.

Oettinger, A.G., with S. Marks. Run, Computer, Run: The Mythology of Educational Innovation. Cambridge, Mass.: Harvard University, 1969.

Roblyer, M.D. "When Is It 'Good Courseware'? Problems in Developing Standards for Microcomputer Courseware." Educational Technology, 21, 10 (1981), 47-54.

Rockart, J.F., M.S. Scott Morton. Computers and the Learning Process in Higher Education. New York: McGraw Hill, 1975.

Thomsen, C.B. "Computer-Aided Instruction." Computers in the Classroom. Eds. J.B. Margolin and M.R. Misch. New York: Spartan Books, 1970, pp. 237-49.

Watson, P.G. Using the Computer in Education: A Briefing for School Decision Makers. Englewood Cliffs, N.J.: Educational Technology Publications, 1972.

THE ENHANCED CAI TUTORIAL

by

Carl B. Jensen, Ph.D.
Department of Psychology
Illinois Wesleyan University
Bloomington, IL 61701
(309) 556-3100

The development of microcomputers has not only brought a new technology within reach of school systems but also has resulted in a significant increase in the number of authors of educational material. With encouragement from hardware venders, publishers have rushed to seek out software programs to list in their catalogs. The result has been a flood of "educational" material that is similar to that produced in the 60s when programmed instruction promised to revolutionize the educational scene. And like the 60s, many educational programs (in this case, courseware) bear only a superficial resemblance to the well constructed programs that are possible, given the state of the art.

In the 60s, programmed instruction was both oversold and underdeveloped. Teachers were either threatened because "it may replace teachers" or dismayed by its disappointing performance. Perhaps educators will soon discount the impact of the electronic age of microcomputers for many of the same reasons which seemed cogent in the age of teaching machines.

Today, educators at all levels of instruction and in almost all disciplines are trying their hands at computer-assisted instruction (CAI) without the benefit of programmed instruction's hard-won knowledge of how to specify, design, and validate good instruction. (Bunderson & Faust, 1976) To date, there are few examples of CAI tutorials which are based on sound learning principles and instructional technology. In fact, "Recent studies have pointed out that with the dramatic fall in the price of equipment, the single most important barrier to instructional computing is the lack of high quality curriculum materials." (Conduit, 1978)

Both programmed instruction and CAI can be effective strategies for the teaching profession, but they can not be routinely misrepresented and be expected to survive in the educational marketplace. We are at the point in the evolution of computer technology, the science of learning, and the technology of motivation where the best of all three fields can and should be combined. These technologies are capable of fostering sophisticated courseware, of advancing CAI as an effective educational tool, and of providing a powerful teaching aid for the classroom.

OBJECTIVES

This project attempts to produce an expanded concept of CAI by applying the instructional technologies that are characteristic of programmed instruction and behavior modification in an experimental design. The project includes not only the courseware, but also an extensive data collection system, a motivation system, and a delivery system for the classroom teacher or parent.

The Tutorial

The objectives of the courseware are to teach basic addition facts up to sums of 20. The courseware determines the entering repertoire of the learner, teaches the unknown addition facts, includes appropriate remedial lessons where needed, develops the long term retention of problems, reviews previously acquired facts, and achieves mastery with less student time and less student/teacher effort than is required by conventional teaching methods.

The Data System

The system of data collection and analysis accomplishes 3 purposes. First, it records each student's responses and constructs a curriculum tailor-made for each student. Second, the data is analyzed and presented in such a way that the teacher or parent is kept informed and in control of the student's learning environment. Third, the data can be used to improve the tutorial itself. The records are of sufficient detail that each session can be reconstructed, thus allowing basic research on the learning process and a detailed component analysis of the tutorial.

The Motivation System

The courseware involves procedures designed to reduce fatigue, to produce immediate knowledge of results, and to maximize success. The courseware not only shows progress, but is entertaining and fun. A complete motivation system for the classroom provides immediate reinforcers (including both privileges and praise), tangible evidence of progress, and a system for acknowledging individual achievements by the peer group, the teacher, and the student's parents.

The Delivery System

The program offers methods to help the teacher administer the educational package to any student. The methods offered integrate the courseware and the motivation system without causing undue disruption in the normal classroom routine. Extensive documentation provides demonstrations, sample classroom materials, and complete instructions. The data presentation informs the teacher of each student's 1) current work, 2) summary of known facts, 3) individualized curriculum, 4) rate of progress, 5) potential problems, 6) cumulative achievements, and 7) work to be completed. In addition, the teacher is provided with a means of demonstrating the tutorial to parents or students. The program gives hints for helping the student through troublesome material, a means of changing the courseware to accommodate students with special needs, and mini lessons to teach rules which make the student's learning easier.

DESCRIPTION OF THE PROGRAM

The Tutorial

The courseware provides a three-way interaction between the teacher, the student, and the microcomputer. After choosing an entering level, the teacher starts the student on the program which tests, retests, then generates an individualized curriculum. In the program, answers are timed to prevent counting on fingers. If a problem is missed twice in a row, it is slated for the teaching sequence. During this sequence, the tutorial always selects the unknown problem with the smallest numbers to be taught next.

If remediation is required, the tutorial initiates it, but if the student is "stumped" and continues to make the same error, the tutorial displays a STOP sign on the screen, locks the keyboard, and calls the teacher. After the teacher unlocks the keyboard with a combination of two keys, the courseware identifies the problem and suggests a solution. When the space bar is pressed, the program resumes where it left off, permitting the teacher to help his student through the troublesome area.

The teacher is also called when the student progresses to a higher level and deserves the teacher's praise for work well done.

When the student is ready to progress to the 9s and the 10s, the teacher is summoned to briefly explain a simple rule for answering all problems with 9 or 10 in them. The teacher is called in instances when a person can best help the learner. The tutorial does the rest.

No reading skills are required of the student and no computer skills are necessary for the teacher to fully implement the total system. Visual aids and prompts are used with each new problem, and prompts are gradually withdrawn as the student is able to do without them. Remedial work is programmed as soon as the student makes a mistake, and is constructed to give just enough prompting to help the student answer correctly again.

To develop the long term retention of problems, known problems are reviewed frequently and are interspersed in a carefully planned sequence with new problems.

The Data System

When the student signs in, the previous day's records are automatically retrieved to formulate the curriculum for that day, thus providing continuity between lessons. When the lesson is over, the data are automatically stored on disk. The data are analyzed in 7 ways for the teacher and in the research version, were detailed enough to reconstruct the entire session. Software is available to analyze the data in still more ways for research purposes.

The Motivation System

Several features of the tutorial contribute to a high work output and desirable study conditions. Frequent restbreaks, immediate feedback on answers, points awarded for each correct answer, and animated displays (such as a happy face which winks, eats problems, and wiggles his nose), are used to reward achievement.

Several techniques are provided to extend the motivation system into the classroom. As students finish each lesson, the problems worked on during that lesson are displayed on the screen. Students then copy the problems onto a certificate, which provides for both transfer from the screen to paper, and a "ticket" for a special privilege like taking the ball out to recess, or passing out papers. A bulletin board of rocket ships, race cars, or a sea scape provides evidence of progress toward sums of 20. Each student's ship or car is moved across lines labeled "2", "4", "6", etc. until the final goal of 20 is reached. A graduation badge can be worn on the day the student reaches sums of 20, and later displayed in a special place in the room as an effective statement of the esteem the graduates have earned.

Indications that motivation is a problem are discussed in the manual and remedies are suggested. One of the data summaries analyzes a measure of progress for each student, thus monitoring and signaling possible problems with motivation.

The Delivery System

A 60+ page manual documents the courseware, the motivation system, and the interpretation of the data summaries. Software provides tables, lists, and graphs of each student's data to summarize strengths and weaknesses, and to document the curriculum development. Several of the summaries can be printed on hard copy.

If academic problems arise in the lesson, suggestions are given for handling them. If technical difficulties arise (like disk full errors), detailed instructions are, again, given in the manual.

The motivation system is explained and samples of teacher-made bulletin board figures are provided for reproduction. Classroom preparation, suggestions for introducing the tutorial to students, and a typical daily routine are described. Finally, procedures are suggested for administering the total package in the classroom which accommodate a variety of teaching styles.

The Development Procedures

The courseware and data system were developed in a laboratory setting with groups of two to four students. Single subject methodology was used to evaluate the courseware during its many revisions.

Once the courseware was validated, it was moved to the classroom setting. Here, the development and validation began for workable classroom procedures, a teacher-oriented data analysis system, a motivation system, and finally, a manual. The development sites included: two first grade classrooms in a University affiliated lab school; two first grade rooms, a second grade room, and a combination second-third grade classroom in a rural farming community.

RESULTS

The major findings can best be described in terms of 1) the effects on the student's academic performance, 2) the student's reaction to the program, and 3) the teacher's experiences in administering the program.

Effects on Academic Performance

The most striking effect is that the students learned the addition combinations to the sum of 20 in about 40 days, working 20 to 30 minutes per day. (The major part of first and second grade math curriculum is to learn addition facts to sums of 20). Figure 1 shows the overall performance of four students in the first semester of their first grade year. The records are typical and show that all of the problems to sums of 20 were learned in 33 days by Katy, in 24 days by Jace, in 34 days by Braydn, and in 31 days by Angie. All of these students began on their first day learning at a level of 0 + 1 = with the highest level being 3 + 3 =. Note that although most students learned only 50 to 60 new combinations, they were ultimately able to answer all 121 combinations.

An examination of the pattern of learning reveals that students usually go through the teaching sequence learning 3 to 6 problems each 20-30 minute lesson. Of those 3 to 6 problems, several are often NEW problems, i.e. problems that the student has never seen before (except in testing). The rest are problems that have been learned in previous lessons but have been missed twice in a row during review and thus are slated for the teaching sequence again. Figure 1 shows the total number of NEW problems accumulated over days. The pattern shows that several NEW problems are learned, with several days then spent relearning twice-missed problems. Once all of the review problems have been relearned, more NEW problems are added to the student's repertoire. Although the progress appears to be in stair-step fashion, the overall progress is steady, as shown by the relatively straight line for all four students.

If the number of NEW problems is computed for an average 30 minutes of work (not including the review problems), students usually learn between 1.8 and 3.2 problems per 30 minutes of work. The four students in Figure 1 show an average of 1.8, 2.9, 2.4, and 2.4 NEW problems for every 30 minutes of work.

Of the 121 combinations (using the numbers zero through 10), the typical student relearns around 8 to 12 of them multiple times, usually 2 to 4 times. If the teacher does not take advantage of the data analysis program and help the student with the particularly troublesome problems, one or two additional problems may be retaught up to 6 or 7 times. Precisely which problems are most troublesome for a large group of students has not been analyzed yet.

The use of rules to short-cut the learning experience shows a significant savings in the student's time. When the courseware was being developed, students who were ready for the 9s were first tested on all combinations with 9 in them. They were then shown and briefly quizzed on the rule for 9s and immediately tested again. Students who had missed a considerable number of 9 combinations in the first test missed almost no problems in the second test. Moreover, if the rule for 9s was reviewed before the next day's lesson, there was no deterioration in the performance thereafter.

Student's Reactions

The student's reaction to the tutorial has been very positive. It was not uncommon for students to volunteer to stay in from recess to take their turn at the tutorial. Even though there was constant noise and movement in the regular classroom, students working on the tutorial did not seem distracted from the screen. Further evidence of high interest was shown by the fact that even though rest breaks occur in the program about every 5 minutes, students elect to spend almost no time in them. Instead they choose to terminate the rest break and move on in the tutorial. This behavior seems to be consistent across virtually all of the students who have worked on the tutorial in spite of the fact that answers are required every 5 to 8 seconds throughout the 30 minute lesson.

The Teacher's Experiences

All of the teachers who have used the CAI tutorial package expressed a variety of reservations prior to using the package. None were familiar with computers and so were apprehensive about using one in their rooms. They expressed concern about administering the tutorial at the same time other activities required their attention, about the distraction the computer might cause to children not involved with it, and about the use of incentives for a small number of students in the class at a given time.

After 2 to 3 days, each of the teachers felt comfortable about the use of the microcomputer in his or her class. After using the package for several months, each teacher was asked for comments about problems and successes they experienced with the tutorial. They said:

> "I could pinpoint individual needs and monitor individual progress. The program's use of teacher intervention for praise and help kept me in touch with each child during his use of the program. I thought it would disrupt the class to have to stop to answer the STOP sign but I found that the computer worked as easily as a learning center and added a dimension to the classroom. It gave me a record keeping system as well as a system to report to parents in terms that we could both understand."
>
> "The positive feedback of what they already know, along with the helpful aspect of where they are having difficulties has been an aid to me in providing individualized instruction. It also provides an excellent opportunity, at conference time, to share with the parents what their child is doing, what he knows and

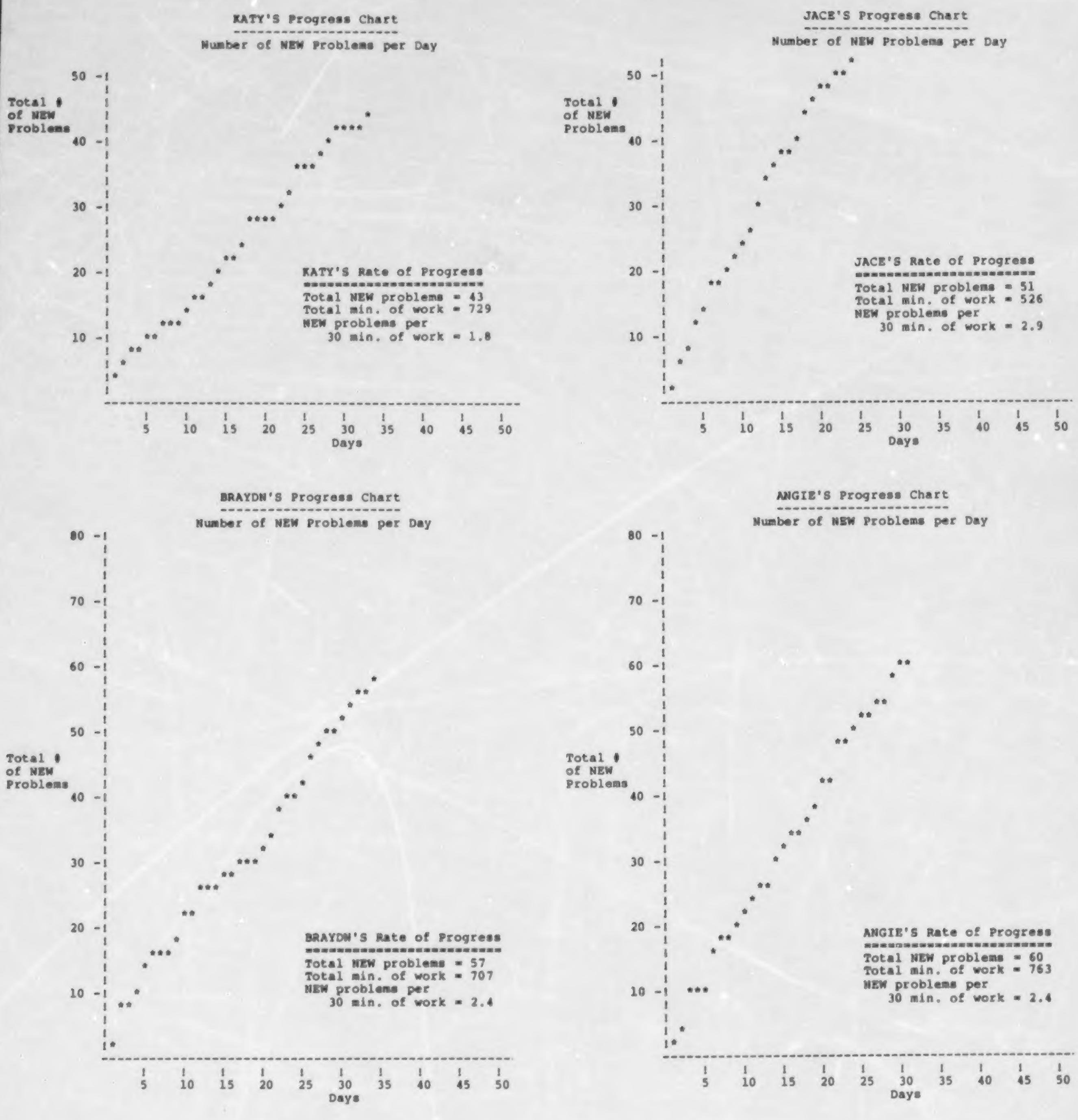


Figure 1. The Total Number of NEW Problems Worked on over Days.

has accomplished, and what he needs to do. My parents were very impressed with the data on their child's work that your program provided."

"Besides being the greatest motivational tool that I have ever used for this area of the second grade curriculum, the program also teaches, reviews, and is an efficient way to keep records for the teacher."

Perhaps the strongest recommendation for the total program is that each teacher asked to use it after the validation procedures had ended.

CONCLUSIONS

The use of an "enhanced" CAI tutorial via a microcomputer in the public school classroom has been a very rewarding experience. Students learned the targeted material in considerably less time than it takes to teach it using conventional methods. That students forget material and must learn it again (with some savings) was demonstrated, but was not surprising. That students consistently worked at the math tutorial for 20 to 30 minutes each day and chose not to end their lesson early or to spend time in rest breaks leads one to reassess the estimate of the attention span for first and second grade children.

The reactions of the students and teachers were very favorable although most teachers expressed initial apprehension before using the educational package.

The results indicate that an "enhanced" CAI tutorial can be an extremely effective way of teaching in the classroom via a microcomputer. Emphasis in this project has been on enhancing the traditional CAI courseware with 1) an experimental analysis of academic performance, 2) a motivation system both on the screen and in the classroom, 3) a comprehensive data analysis system, and 4) a delivery system to aid the teacher in its administration.

REFERENCES

Bunderson, C. Victor, and Gerald W. Faust. "Programmed and Computer-Assisted Instruction." The Psychology of Teaching Methods. Chicago, Illinois: The National Society for the Study of Education, Distributed by the University of Chicago Press, 1976.

Conduit. What is Conduit? University of Iowa: Conduit, 1978.

Computer Services

GOVERNANCE AND POLICY REGARDING COMPUTER SERVICES AND INFORMATION SYSTEMS IN SMALL TWO-YEAR COLLEGES

Lora P. Conrad
Route 5, Box 355
Russellville, AL 35653
(205) 332-4627 or 993-5331

BACKGROUND

The role of computers in the small two-year college, and therefore, the role and responsibilities of the people in charge of the computers is changing rapidly. With increasing data demands by state and federal authorities, decreasing budgets that demand more efficient operations, and decreasing hardware costs, more and more small colleges are turning to computer services and information systems in some form for assistance. Data processing is such a new field that job definitions, qualifications, and professional standards are in the same state of flux as the hardware technology. Data processing within the administration of a small college is even newer. As a result, little information is available on the role of computer-based systems and associated staff in governance and policy at the small, public college.

The purpose of this study was to determine the current status of computer services, the role and the method of governance of computer-based information systems in small, public junior and community colleges and the policies affecting those systems. Specifically, all accredited public two-year colleges with membership in the American Association for Junior and Community Colleges (AACJC), of less than or equal to 2,000 headcount as of fall 1979, were surveyed regarding governance and policies affecting computer-based information systems and the associated computer services. The results were obtained by tabulating and analyzing the survey responses so as to answer a series of questions regarding these issues.

The results of the study can benefit any small college that has or plans to have a computer-based information system to provide administrative and other services by defining the typical status and common governance and policy directions affecting similar colleges. The study can serve both as a planning guide and a comparative reference for the President of a small college seeking information regarding the most common practices, procedures, and policies regarding computer-based administrative information systems in similar colleges.

A review of the literature finds primarily studies of much larger institutions and their information systems, even the CAUSE proceedings reviewed discussed primarily colleges of 4,000 or more in the Small College Track (Schouest and Thomas: 1978) (NagSiney and Thomas: 1979). The Directory of Computing Facilities (Darr: 1977) did not specificaly identify the small public two-year colleges, although they were included in the directory. The issue of whether or not small colleges need their own computer services and administrative information systems is not even settled as far as the literature is concerned. As recently as 1979, St. John described a combination manual/automated system as appropriate for colleges with up to 1,800 full-time equivalent students.

PROCEDURES

The procedures utilized in conducting this study were as follow:

- Determined institutions to survey.
- Developed survey instrument.
- Evaluated survey by sending to a sample.
- Revised survey.
- Mailed survey to all colleges selected.
- Followed-up on non-returns.
- Tabulated results using a computer-based procedure.
- Analyzed and reported results.

The study included all public junior or community colleges that are accredited and had a 2,000 or less headcount in fall 1979. No minimum headcount figure was set. This figure of 2,000, though arbitrary, is an often used size parameter in discussions of colleges and was adopted as a valid cutoff point in defining "small" colleges. The AACJC Directory (1980) was used as the source of data both for defining the study set and for obtaining mailing addresses. This analysis produced a study set of 287 institutions. Concurrently the survey instrument was prepared, evaluated, and revised. (Conrad: 1981). The survey along with a letter of explanation was forwarded to the 287 colleges. A follow-up letter was sent to all institutions that had not responded several weeks later.

The survey form was designed to obtain responses from two groups:

- Institutions with computer services.
- Institutions with no computer services.

For those institutions that have computer services available, the following questions were analyzed:

- What is the role of computer services and information systems?
- What is the scope of the services provided?
- What is the level of conceptualization and implementation of information systems?
- Whose responsibility is data base integrity and validation?
- What are the staff positions and numbers?
- What proportion of the computer services budget is for personnel?
- To whom does the head of computer services report within the organizational structure?
- What is the authority and jurisdiction residing in computer services?
- What are the policies affecting hardware, software, and/or services acquisition?
- Who evaluates the computer services and information systems?

For those colleges currently without computer services, their plans, if any, for computer services were assessed.

The following assumptions and limitations may affect the results of this survey:

- The survey respondents were assumed to be representative of the entire study group.
- The data provided by each respondent was assumed to be complete and accurate.
- The survey instrument developed assumed a basic knowledge of computer and information systems terminology by the respondent.

RESULTS

From the initial group of 287 small colleges contacted, a total of 172 colleges or 60 percent responded. Of these, 165 or 57 percent of the initial group provided surveys that are included in the analysis. Of the 165 colleges, 80 percent or 131 reported some type of computer services; 20 percent or 34 reported no computer services; of the 34, five have no plans at all for computer services; that is 3 percent of the colleges participating in the survey have no plans for computer services.

Survey results for the 131 institutions having computer services were tabulated and analyzed as percentages. Where institutions gave multiple responses, 131 was used as the percentage base; where less than 131 responded, percentages were calculated on the number providing a response to the question under review. Results were analyzed for those that did not have computer services to determine their plans for hardware and software. Each study question and a synthesis of the responses are given below.

What is the role of computer services and information systems?

To assess this role, the colleges were asked to prioritize the functions of computer services from a list of seven major areas: Student Information Systems, Financial Systems, Other Administrative Support, Computer Science Instruction, Instructional Support, Computer Assisted Instruction (CAI), and Library Support.

Top priority, therefore, the primary role of computer services, was assigned by 43 percent of the colleges to Student Information Systems. Fully 39 percent of the colleges gave instruction in computer science or data processing as top priority. Another 25 percent gave Financial Systems top priority. Small colleges do not yet give significant roles to computer-assisted or computer-managed instruction, as about two-thirds of the respondents did not give either any priority ranking among the seven areas. The services provided are an average of 61 percent administrative, 35 percent instructional, 1 percent leased to agencies outside the college, and 3 percent other services such as community service.

What is the scope of the services provided?

To assess the scope of services offered, each major system as listed above was subdivided into commonly needed subsystems which were noted as available, needed, or not ranked.

The most commonly available subsystems are in the Student Information System ... 80 percent have registration and add/drop (course changes) computerized and 72 percent have computerized application processing and maintain a student master file. Student grade analysis is computerized by 69 percent; computerized transcripts provided by 58 percent.

The next most commonly available system or support is the Financial Information System, although only 36 percent have complete financial reporting. Payroll is computerized at 62 percent of the colleges with standard financial reports such as subsidiary and general ledgers available at 50 percent of the colleges.

A total of 24 percent of those studied regard their information system as an integrated Management Information System. An additional 25 percent regard their information system as several comprehensive subsystems. Another 10 percent regard their system as a Transactional Information System, but fully 47 percent rate their system as only a collection of separate applications. Only 8 percent are now upgrading their system from one level of software implementation to the next.

Whose responsibility is data base integrity and validation?

Data validation and data base integrity are primarily the responsibility of the user in 59 percent of the colleges but the responsibility of computer services personnel in 42 percent of the colleges. (Two colleges assigned both the responsibility).

What are the staff positions and numbers?

An average of 2.74 full-time equivalent staff work directly for computer services, with total staff sizes that range from one to nine people. Computer services is, in fact, a one person office in 26 percent of the colleges, however two people are most frequently employed, as reported by 28 percent of the colleges.

The position reviewed, the average full-time equivalent (FTE) staff in that position, the range of numbers of people reported in that position, and the percent of all employees each type represent are as follows:

Position	FTE	Range	Percent
Analyst	.16	0 to 1	7%
Programmer	.3	0 to 2	16%
Computer Oper	.59	0 to 2	26%
Secretary	.20	0 to 2	9%
Data Entry	.73	0 to 9	32%
Other Titles*	.26	0 to 2	10%

Primarily 1 to 2.
* Includes instructors and student aids.

What proportion of the computer services budget is for personnel?

The average annual budget reported for computer services is $59,957, including hardware, lease, software, personnel, supplies, and support expenses as reported by 115 colleges. The average personnel cost as reported by 92 colleges is $30,991, or 52 percent of the average budget.

To whom does the head of computer services report within the organizational structure?

The largest single group, 44 percent, report to the chief business officer, 21 percent report to the student services officer, 13 percent to the chief academic officer, 10 percent to the president, and 6 percent to the development officer. A total of 14 percent report to other officers such as a department or division chairperson. Fully 8 percent indicated the person reported to two or more of the officers noted above.

What is the authority and jurisdiction residing in computer services?

The head of computer services is primarily at the third level of management, that is, reports to administrators that report to the president in 76 percent of the colleges. The position of head of computer services is regarded as administrative in 53 percent of the colleges studied, as professional support in 37 percent, and as instructional in 17 percent. The position is regarded as dual--usually both instructional and administrative--in 7 percent of the colleges.

The head of computer services actively participates in setting policy regarding computer services in 85 percent of the colleges. However, that person does not participate in policy making outside the computer services area in 59 percent of the colleges studied.

The authority or responsibility of the head of computer services includes establishing the computer services budget in 40 percent of the colleges, selecting hardware and software in 32 percent of the colleges or assisting with that selection in 55 percent and evaluation of computer services in 23 percent of the colleges.

What are the policies affecting hardware, software, and/or services acquisition?

Such policies can be categorized as 1) the state provides hardware, software, and services, 2) the state regulates hardware and/or software procurement, 3) the college sets such policy independent of specific state regulation, or 4) a combination of these.

The state provides computer services to 37 percent of the colleges. The state regulates hardware purchase in 44 percent of the colleges. The colleges set their own policies for hardware, software, and computer services in 53 percent of those responding.

However the policy decisions are made,

actual selection of hardware and software is made by a variety of people, usually by several different offices within a college. These include the head of computer services in 32 percent of the colleges, a special committee in 25 percent, the college executive council in 30 percent, the business officer in 18 percent, and the president in 16 percent.

The policy of the colleges in regard to obtaining access to hardware has been to buy hardware in 62 percent of the colleges, to share a system with another college in 18 percent of the colleges, and lease time at other locations in 15 percent of the colleges. Other methods of obtaining services were used by 23 percent, including state provided and college lease-purchase. Hardware was frequently purchased even if access was available to a state system. These data indicate that fully 18 percent utilize multiple methods of hardware access.

The policy regarding software acquisition can probably best be described as obtaining software anywhere it can be found. Multiple sources were used in most cases. Although 58 percent have developed their own applications as needed, 40 percent use software developed by the state or other in-state colleges, 22 percent use applications purchased from outside software developers, and 19 percent purchased applications from the hardware vendor.

Who evaluates computer services?

Computer services are not formally evaluated at 23 percent of the colleges. In 19 percent of those evaluated, two different methods or groups conduct the evaluation. The head of computer services provides evaluation in 23 percent of the colleges; the administrator over computer services provides evaluation in 39 percent of the colleges; a users' committee provides evaluation in 28 percent of the colleges, and 6 percent use other methods.

What are the plans of colleges presently without computer services?

Of the 34 colleges currently without computer services of any type, five had no plans. For the remaining 29, their plans for meeting hardware and software needs were analyzed. Of these, 41 percent expected to procure their own hardware, 17 percent expected to share hardware with another college, 10 percent to join a state system, 7 percent to lease a system, and 7 percent to lease time on a system. Only 24 percent were undecided as to how to meet hardware needs (primarily those who do not plan to obtain computer services for over three years). To obtain administrative information systems software, 38 percent expect to obtain such systems from other colleges, 34 percent to develop their own, 24 percent from software companies, 10 percent from the state, 7 percent from hardware vendors, and 21 percent were undecided.

DISCUSSION

The policy of small colleges of less than 2,000 headcount is primarily--62 percent--to buy computer hardware to meet their computing needs. This differs from St. John's (1979) recommendation that colleges of less than 1,800 FTE, (which almost all the study colleges are) should use contractual arrangements with other agencies, either institutions or businesses to meet their data processing needs. Plourde (Schouest and Thomas: 1978) identified several computer system options for small colleges. These include consortia which is consistent with this study which found 18 percent sharing a system with another college in some way. Another option Plourde identified was minicomputers with owned software, which this study found most often in that 62 percent bought hardware and 58 percent developed some or all of their own software. For the option he identified of a minicomputer with leased or contracted software, this study found 19 percent obtaining software from the hardware vendor and 22 percent obtaining software from software developers. Proprietary software and hardware--so called turnkey systems--were also seen as a small college option by Plourde, however, very few such systems were identified as in use in the small college.

Nicely (Schouest and Thomas: 1978) recommended that commercially available software be investigated prior to development which is being done in approximately 41 percent of the colleges. Little mention was found in the literature regarding use of software developed in-state by other colleges or the state, yet this is a major source of software, being one of the software sources in 40 percent of the colleges.

The low level of state regulation regarding software (14 percent) is interesting in light of the rising cost of software--especially the expense of software development.

The budget distribution of computer services among personnel, hardware, and other items for small colleges is very similar to that in the literature for larger institutions. At the small colleges studied, 52 percent of the average budget is for personnel. Kehl (1979) reported an average of 54 percent for personnel salaries at the thirteen universities he studied.

The role of the head of computer services in policy making at the college ran the gamut of none--not even regarding computer services in 15 percent--to policy making in computer services and also in areas outside computer services in 41 percent of the colleges. This relatively large proportion with wider policy input is all the more interesting since only 10 percent of these individuals report to the president directly. Though Richman (1974) regards Data Processing as the fifth most powerful college subsystem, the overall profile of the head of computer services that emerges from this study is not of a person having that level of power or authority within the small college.

A single computer system is most often used for all purposes in the small college--both administrative and instructional. Though 61 percent is administrative, instruction in computer science or data processing was the second priority using 35 percent of the computer services provided. Most interesting is the fact that this is for teaching about computers not teaching with computers. CAI seems to have both a low implementation and a low interest level, though this might be affected by the fact that the survey respondent was not usually an instructor.

Although almost all colleges had some elements of a Financial System and a Student Records System, less than 50 percent of the colleges had other administrative support systems such as Academic Information, Management and Research, Financial Aid, Follow-up, Inventories, and Library support, except for the subsystems of enrollment analysis and instructor load. Interest in such systems as Academic Information, Management Information, and Financial Aid was about 40 percent indicating a tremendous need for suitable software at the small college in these areas.

RECOMMENDATIONS

From the results of this survey, the review of the literature, and practice, the following recommendations are offered to presidents and administrators of small, public, two-year colleges regarding computer services and information systems.

- Presidents have not fully recognized the importance and power of information systems. It is recommended that the potential power of information systems be carefully considered when placing computer services and information systems within the organization.
- Small colleges are acting almost opposite to literature recommendations in seeking, by a large majority, to purchase hardware. This approach is often selected from a naive perspective regarding long term costs. It is recommended that small colleges review the options available to them more carefully, with the expert assistance of someone not employed by a hardware vendor, prior to making a decision.
- Small colleges are wisely seeking software from many sources, rather than planning development. It is recommended that assistance be sought first through other public colleges within the same state. Effective use of a computer requires appropriate software, which can most likely be located at similar, slightly larger colleges.
- Small colleges are attempting to provide computer services with primarily operators and entry-level personnel. Due to pay scales, lack of development work, and often not the "latest" hardware, keeping experienced people is difficult. This means location and purchase of suitable, documented software should be a high priority rather than in-house development, and that attrition should be anticipated.
- Combining administrative and instructional applications on one computer system is a practical reality for small two-year colleges. A dual position for the head of computer services however, means one support area or the other will suffer. It is recommended that the head of computer services be an administrative position, with instruction in computers lead by someone else, if possible.
- Though CAI/CMI are not yet major features in the small college, they will occupy a growing role. (Historically small colleges have lagged industry and larger colleges in computer applications). It is recommended that small colleges begin introducing CAI/CMI concepts by embarking upon a "computer literacy" effort for faculty, that potential CAI/CMI applications be included in long-range planning, and that both microcomputers and expansion of current systems be considered to meet those needs.

REFERENCES

American Association of Community and Junior Colleges. 1980. "Community, Junior, and Technical College Directory."

Conrad, Lora P. 1981. "Computer-Based Information Systems Policy at Small Community Colleges." Ft. Lauderdale, FL: Nova University.

________. 1981. "Governance of Computer-Based Information Systems in Small Community Colleges." Ft. Lauderdale, FL: Nova University.

Darr, G. 1977. Directory of Computing Facilities in Institutions of Higher Education Throughout North America. Annual Seminar for Directors of Academic

Computing Services, Eugene, OR: Oregon State System of Higher Education.

Kehl, W. B. (Editor). 1979. Profiles of Academic Computing. Seminar for Academic Computing. Corvallis, OR: Oregon State University.

Naginey, Charles H. and Charles R. Thomas (Editors). 1979. Planning Higher Education Information Systems for the 1980's. Bounder, CO: CAUSE, November, 1979.

Richman, Barry M. and Richard M. Farmer. 1974. Leadership, Goals, and Power in Higher Education. San Francisco, CA: Jossey-Bass.

Schouest, Michael R. and Charles R. Thomas (Editors). 1978. Development, Use, and Management of Information Systems in Higher Education. Boulder, CO: CAUSE, December, 1978. (ED 171 228).

DEVELOPMENT OF AN ACADEMIC SOFTWARE MANAGEMENT SYSTEM IN A SMALL COLLEGE ENVIRONMENT

Thomas Moberg and Mary Verkade
Grinnell College
Grinnell, IA 50112

Thomas Dahl
33 Main Street
Framingham, MA 01701

A. INTRODUCTION.

The organization of computer software at small academic institutions can be a difficult and time-consuming task. Academic software used by faculty and students includes vendor-supplied programs, materials written by local users, "borrowed" programs, commercial packages, databanks, and all the forms of documentation which describe the programs. The software items might reside on disks, diskettes, magnetic tape, paper tape, cards, or paper listings.

Computer systems generally have some vendor-designed account structure which provides a vehicle for organizing software on the system, but the computer center staff may have considerable latitude in using the existing structure. Software might be tucked away or dumped into all sorts of accounts in the computer memory system, with no particular organizational scheme. Good academic software may exist in general accounts, faculty accounts, departmental accounts, student accounts, and sometimes in forgotten accounts. While the computer center staff may not be charged with organizing and maintaining academic software libraries, it is obvious that such libraries can contribute in a significant way to a high-quality computing operation.

When the software which academic users want and need is poorly organized, several unpleasant consequences result. Users are frustrated when they cannot find a particular program, computer center staff members waste time hunting for things, programmers rewrite programs which already exist, updates and modifications to programs are hard to do systematically, and materials which might be useful pedagogically are ignored or forgotten.

This paper describes the development of a software management system designed to alleviate the problems caused by disorganized and undocumented software. The history of the original and present versions of the software cataloging system is given in Section B. Section C contains a description of the general structure of the present software management system (Softman) while Section D presents the various features in more detail. The internal structure of Softman is described in Section E, and Section F contains a summary of operating system requirements and information on the availability of the software management system to other installations.

B. DEVELOPMENT HISTORY OF THE SOFTMAN PROJECT.

Grinnell faculty and students began using computing in the mid 1960s, first using a batch system and, since 1975, via interactive terminals connected to Digital Equipment Company (DEC) PDP 11 computer systems. By 1976 it was obvious to the computing staff that the academic software was very disorganized and that all the unpleasant consequences mentioned above were being experienced. It was clear that some systematic method of identifying and cataloging software was needed, and between 1976 and 1979, the first cataloging system was developed. When completed, the system consisted of an on-line data base, a printed catalog for distribution, and auxiliary programs to construct an index and to permit searching the data base for particular items.

The first step in organizing the software was to identify the items, describe their characteristics, and install them in appropriate locations on the computer system. Items were classified by discipline area (e.g. mathematics, biology) or by special features they shared (e.g. simulations, graphics programs, databanks, utilities). A separate computer account was used to store the items of each particular type. For each software item, the location, name, author, documentation available, and a description were recorded.

The software data base, with items organized according to the accounts in which they resided, was constructed using text-editing programs. Whimsically dubbed PAPA.DOC by Don Renfrew, a Grinnell College student who worked on the on-line catalog in the summer of 1977, the original version contained entries for over 450 software items. Printed copies of the catalog file itself were distributed to computer users.

A program to produce an index for the catalog was written during the summer of 1978 by Carolyn Fieg, a Grinnell College student. The printed catalog file with index appended totalled 125 pages.

The program used to search the catalog for specific items was written by an academic programmer, Robert Christiansen, in early 1979. This essentially completed the cataloging project as originally conceived. New items continued to be added, old items revised, and regular versions of the catalog printed for users. Printed copies of the catalog were distributed to many other academic institutions and copies of the cataloging programs were given to several colleges.

Although PAPA.DOC made a significant contribution to academic computing at Grinnell College, a number of weaknesses in the system became apparent as the computer operation expanded. It was difficult to update the catalog, and there were stringent limitations on the categories available and the amount of information which could be included for each item. In the spring of 1981, the academic computing staff reluctantly decided that a major change was needed. As there was no commercial software available at Grinnell College at the time which could satisfy the cataloging needs, a decision was made to design a package locally.

After discussions with academic users, the academic staff drew up general specifications for a new system. An outstanding student programmer, Tom Dahl, was hired to write the system during the summer of 1981. Most of the internal design features of the system were developed and programmed by Dahl, and after he left Grinnell in September 1981, Programmer-Analyst Mary Verkade cleaned up the loose ends and installed the package.

The general purposes of the new Softman system were to:

1. provide a complete catalog of all publicly accessible software on the computer system, organized in a way which would allow users to do an on-line search for particular items;
2. serve as a master record-keeping system with each item described by a large number of salient features for use by academic programmers in classifying, implementing, documenting, and updating programs.

Transportability of the package was also considered to be a significant design consideration. Few small colleges have the staff or resources to develop major systems of any type, so it is very important for such institutions to share as much as possible. Thus, the Softman package was designed to be implemented with minimum difficulty on DEC PDP 11 computer systems similar to the one at Grinnell College.

C. OVERALL DESIGN OF THE SOFTWARE MANAGEMENT SYSTEM.

The Softman system may be used to reference any number of data bases. It is assumed, however, that a single, primary data base comprises the public software catalog. As it is this data base which is of interest to the general user, it is designated as the default data base for interactive purposes. For specialized uses, additional data bases may be included by the staff at any time.

A data base consists of accounts, items, and components. All accounts for which there are software items to be included in the catalog are itemized in the data base. Within each account, a variable number of items, usually programs in executable image, are defined. An item may be a single, stand-alone program or an integrated package consisting of a number of individual modules which may or may not be capable of independent execution. In the latter case, each module is referred to as a component.

For each item or component, thirty-nine information fields are available. Seven of these fields are fixed in that their definitions are explicitly specified in the code and may be changed only through programmer modification of the source. The remaining thirty-two fields are dynamically defined and may be altered or deleted interactively at any time without programmer intervention.

The seven explicitly defined information fields include:

1. the name of the item;
2. a list of available documentation for the item;
3. a short description of the item;
4. brief instructions on using the item such as the CCL if one is defined (CCL = Concise Command Language; a DEC facility by which a program may be run via a one word command rather than explicitly specifying "RUN program name");
5. a list of keywords;
6. the number of associated items;

7. the number of components. Components were defined above. An associate is another item integrally related to the current item but not a component of that item. Items may reside in the same account as their associated item(s) or they may reside in different accounts.

Fields 1 through 4 are of most interest to the general user. The remaining fixed fields and, at Grinnell College, the thirty-two variable fields hold information of particular interest to maintenance programmers and other Computer Services staff. The variable fields may include such information as:

1. the source or author of the item;
2. location of the executable image;
3. location of the source program;
4. programming language in which written;
5. person responsible for most recent implementation;
6. inclusion of standard CCL (see definition above) entry;
7. recommended restrictions to access if any;
8. specific hardware requirements;
9. degree of support committed to the item;
10. references in code to specific hardware items;
11. use of PEEKs;
12. inclusion of standard security procedures;
13. use of specialized hardware instructions such as keyboard cursor control;
14. form of referencing external files;
15. list of external files accessed;
16. use of undocumented system features;
17. guidelines for distribution of item;
18. deficiencies as defined by the currently adopted software standards.

D. FEATURES OF THE SOFTWARE MANAGEMENT SYSTEM.

The software cataloging system is designed to satisfy the needs of both general computer users and Computer Services staff members. For the general computer user, the cataloging system provides the following features.

1. Current information on all public software is accessible on-line. Information concerning software items need never be outdated because of the ease and simplicity of updating the data. The user is not tied to a paper copy which may be outdated, easily misplaced, or forgotten. Commands such as SELECT, SHOW, LIST, DISPLAY, and UTILITY OUTLINE allow the user to examine desired portions of the data base.

2. Selection of software items to satisfy particular needs is facilitated by on-line search capabilities. Through the FIND command, a user may search for a specific string or strings in particular information field(s).

3. Up-to-date printed copies of the data base including introductory explanatory notes and complete index are available free of charge to faculty and student users and may be obtained either from the Computer Services resource area or by user generation. Fully formatted and indexed copies are easily generated by the RUNOFF command provided in the package. The RUNOFF command may be executed by the general computer user as well as by staff members. Thus, the printed copies need never be far out of date, and together with on-line availability of the data base, provide convenient reference sources.

4. Clear, concise, informatory remarks and user queries with ample on-line help assist the general user in locating software items. The friendly, jargon-free user interface is especially helpful to new users. Help while using the on-line data base may be obtained in various ways. Responding to a prompt with only a carriage return or a question mark invokes a short explanation of the type of information the user is to enter. Entering "HELP" to a mainline prompt generates a brief explanation on how to obtain assistance for particular commands, while "HELP 'command'" provides the user with information about the command specified. A BACKUP command is provided to permit the user to return to the previous query or, alternatively, the user may generally accomplish the same action by entering control/Z. EXIT or control/C exits the program. (Control/Z and control/C, which utilize the "control" key in conjunction with "Z" and "C," respectively, are standard sequences in DEC Basic Plus.)

For Computer Services personnel, the package assists in identifying software items which require special attention during:

1. updating of the operating system;
2. alteration of hardware configurations;
3. significant modification to or removal of an item on which other items may depend;
4. upgrading of coding standards;
5. restructuring of security policies;
6. maintenance of the item;
7. distribution of the item.

Through the FIND command, a staff member may locate all programs which have a particular characteristic in common. For example, when the DEC editor EDT was introduced at Grinnell College, a number of programs began to behave strangely. These programs all used direct cursor addressing through the execution of certain ESCAPE/character sequences. However, the correct execution of the ESCAPE sequences depended on the keyboard being set in a particular mode, whereas when a user exited from the editor the keyboard mode was changed to one which was incompatible with the direct cursor addressing ESCAPE sequences. It was necessary to modify the affected programs so that when execution began, the keyboard was reset to the compatible mode. Including information of this nature in the variable information fields greatly facilitates the identification of problem areas in software, possibly even before malfunctions are observed. The challenge in achieving maximum utility of the information fields is to be able to foresee tomorrow's software problems which may occur as a result of many factors such as an updated operating system, changes in hardware configurations, or adoption of other software.

Scheduling maintenance and setting priorities for upgrading the public software, which are of great importance where human resources are limited, may be expedited through appropriate use of the variable information fields. The software management system can also provide information concerning weaknesses of individual programs and help articulate and focus the current software standards in existence at an installation.

The Softman package permits easy updating of information fields by clerical personnel without the need for programmer intervention. The ADD and DELETE commands permit insertions and deletions of data bases, accounts, or software items. Through the CHANGE command, the contents of any of the fixed or variable fields for an item may be modified. The information collection and data entry aspects are by far the most time-consuming parts of the cataloging operation. However, once the backlog of existing programs has been handled, it is a simple matter to add information on new programs to the catalog concurrently with implementation of the programs on the computer system.

Variable field designations may be modified, added, or deleted through the ALTER command. The DEFAULT command changes the definition of the default data base and the INCLUDE or EXCLUDE commands may be used to include or exclude data bases from the list of data bases referenced by the software management system.

E. INTERNAL STRUCTURE OF A DATA BASE.

Each data base defined for the software management system occupies an individual file on disk. The internal structure of a data base is organized as a linked list of records. Links are pointers to the physical location of the information in the file including the block and byte numbers where the data begins and the length in bytes of the data. A header block starting in the first byte of the file contains:

1. a code flagging the file as a Softman data base;
2. the creation date of the data base;
3. links to the title of the data base, the description, the first free block in the file, the first free byte in the first free block, and the first account in the data base;
4. the number of accounts in the data base.

Similiarly, each account record begins with a code flagging it as an account record followed by the name of the account; links to the title, description, first item in the block and the next account record; and the number of items in the account.

An item record follows a similar structure with a flag marking it as an item record, followed by links to the item name, instructions for using the item, the documentation directory, and item description. Also included are a link to the first component item, if any, and a backwards link to the item's parent, if it is a component of another item, plus links to the first associated item (if any), the root of the keyword list, and the next item, and a backwards link to the item's account. Completing the item record are the date of implementation, the most recent modification date, the number of associated items, the number of component items, and links to the variable information fields for the item.

Deletions of data base entities are accomplished simply by breaking the links. Thus, the space occupied by the deleted entity is unusable. A REORDER command is provided in the package to optimize the data base structure and eliminate unused space.

A DUMP utility is also provided in case the data base becomes corrupted, as for example, may result from a system crash during an update of the data base. The DUMP utility maps the blocks of the data base so that the links may be traced.

F. OPERATING SYSTEM REQUIREMENTS AND AVAILABILITY OF THE SOFTWARE MANAGEMENT SYSTEM PACKAGE.

The Software Management System was written using DEC RSTS/E Version 7.0 Basic Plus. The approximately one-thousand block package consists of a driver and a group of appendable routines which are combined into seventeen individual modules. Each module runs at less than 16K. Copies of the source code for Softman are available from Grinnell College Computer Services by sending a tape and twenty dollars for processing and handling.

LINKING MICROCOMPUTERS TO TIMESHARING SYSTEMS: EFFICIENT COMPUTING IN A LIBERAL ARTS ENVIRONMENT

Lynn Arthur Steen
CAUSE Project Director
Department of Mathematics
St. Olaf College
Northfield, Minn. 55057
(507)663-3114

The 2000 or so liberal arts institutions in the United States pride themselves on offering a versatile intellectual background for subsequent careers and professional work. In this age, computing must be part of that base, for students of every major. St. Olaf College, Minnesota's largest private liberal arts institution, has linked microcomputers to its timesharing system as a means of spreading computing power throughout the campus at relatively low cost. In the two years since this linkage was created, the number of computer stations available to faculty and students has nearly doubled, and the use of computing has spread to every division in the College.

In this paper we will focus on one key aspect of the use of microcomputers at St. Olaf College: their links with the central timesharing system, and the impact of these links on the development of educational applications. Our focus is on the appropriate and efficient use of technology, on systems that match well the needs of students and faculty with the power and cost of computing equipment.

Much of the work reported on below was supported by an NSF CAUSE grant to St. Olaf College, and other parts have been supported by an NSF/NIE grant for the development of prototype computer materials for teaching secondary mathematics. The major focus of the CAUSE project was to expand the role of computing, statistics, and applied mathematics throughout the mathematics and science curricula.

GENERAL OBJECTIVES

Computing in a college environment is at its best when it is a community activity. A college, after all, is a community devoted to learning and teaching. So computing at a college should serve these ends, helping all members of the community in their common endeavors. It should help those who teach and those who learn; it should serve both the arts and the sciences; it should present new horizons as well as speed the academic journey to present objectives.

Historically, the two most important innovations that helped join computing to the academic endeavor are timesharing and Basic--both products of one of the nation's leading liberal arts institutions, Dartmouth College. Timesharing brought computing power to the entire academic community, while Basic provided a common language of computer discourse. While computer scientists may argue about the merits of Basic as a programming language, it is without doubt the single most important influence in the spread of computing knowledge in the two decades since its invention.

The recent sudden infusion of personal computers in the marketplace provides new opportunities in academic computing comparable to the invention of timesharing and the development of Basic. Powerful microcomputers now cost what terminals alone cost just ten years ago. What Basic did to introduce programming to a generation of students, home computers are now doing at a much earlier age. What the timesharing system of the seventies did for academic computing, small personal computers will do in the eighties: offer ready access to computing for all faculty and students.

On a college campus the transition from timesharing systems to microcomputers also represents a shift from centralized planning to decentralized decisions, from the campus computer center to departmental computer rooms. With decentralization comes incipient chaos: as each professor or department chooses a different microcomputer, the community focus afforded by timesharing is weakened. Soon academic computing becomes a new tower of Babel.

Foreseeing these risks at the transition from the timesharing decade of the seventies to the personal computer decade of the eighties, St. Olaf College embarked on a microcomputer development project designed to preserve the common advantages of timesharing along with the diverse strengths of microcomputers. Our goal was for microcomputers to serve the entire academic community--in teaching, learning, and research--with the same common focus as the timesharing system. Although our experiment in coordinating campus microcomputing is far from complete, we have learned enough to feel that the accomplishment (and frustrations) are worth consideration by other institutions confronting the same issues. While the specific details of our experience will not transfer directly to other institutions, the general objectives and methods should provide an important example for those in similar circumstances.

SATELLITE COMPUTERS

One fundamental principle of academic computing at St. Olaf College is that microcomputers should function as satellites to the timesharing system. Some microcomputers are linked by wire to the timesharing system, both so that they can function as ordinary terminals and so that files can be transmitted from one to the other. Other microcomputers--now the majority--are on a looser tether, being linked to the general system primarily via transfer of floppy disks and, occasionally, by phone modem. The massive storage and processing capability of the timesharing system--essential for many classroom computer assignments--counterbalances the centrifugal tendencies of the microcomputers to help keep them in the campus computing orbit.

Another objective of computing at St. Olaf is that academic computing should serve humanities applications as well as scientific needs. Not only is this the only appropriate goal for a liberal arts institution, but it is also a politically effective position in terms of long range budget support. In practice, it means that such applications as text editing are supported on a par with statistical analysis and numerical calculations. Academic computing, like the library, should serve the entire academic community.

In practical terms these two objectives reinforce each other. Since a major part of academic computing involves word processing and elementary programming, both of which are keyboard intensive, the satellite microcomputers can relieve the timesharing system of a significant yet simple-minded load. This saves the more powerful central computer for the complex tasks it can do so much faster than the microcomputers. Moreover, the satellite linkage of microcomputers to timesharing makes possible a common software basis for campus computing, thus reinforcing within the computing environment the community focus of teaching and learning at a liberal arts institution.

HARDWARE

Generalities and objectives take shape in specific hardware and software decisions. At St. Olaf we use a PDP 11/70 to provide timesharing academic computer services to about 3000 students, 200 faculty, academic secretaries, and other support staff. (Administrative computing is done on a separate IBM system, staffed independently of academic computing.) The PDP 11/70 has one megabyte of memory, and two 300 megabyte disk drives for on-line storage of files and data. There are approximately 60 terminals and a dozen other devices controlled by the 11/70 through its 64 I/O ports. Most terminals and devices are available for general use by any student or employee of St. Olaf College, 16 hours a day, 7 days a week. As library privileges are available to any member of the St. Olaf community, so are computer privileges. Any person can use the general accounts and any student or staff member can obtain a private account just by asking.

The PDP 11/70 runs the UNIX operating system, a paradigm system that provides users with excellent tools for file manipulation. UNIX provides a superb environment for students to learn the principles of computer operating systems. Moreover, because it is written in the relatively high level language C, our system manager can modify it as necessary for special purposes. This ability has turned out to be of crucial importance in the microcomputer project, since slight modifications to parts of UNIX were necessary in order to complete the linkages between UNIX and the satellite microcomputers.

Our choice for microcomputers--a decision made two years ago--is the Heath/Zenith model H89 (now H90). Our criteria for selection of a microcomputer included items such as:

- -- Unitized construction (for portability), including floppy disk storage;
- -- Full ASCII keyboard, with professional (IBM-style) layout for touch typists;
- -- 80 x 24 screen display;
- -- Z80 microprocessor and CP/M capability for maximum access to available microcomputer software;
- -- Total access to hardware and software details, both for system maintenance, and in order to customize the microcomputer system to our PDP 11/70 UNIX requirements.

We run the H89 with two operating systems --HDOS, the Heath Disk Operating System, and CP/M--and can make available to campus users software available under either system.

A similar decision made today would be based on slightly different criteria (perhaps emphasizing 16 bit rather than 8 bit microcomputers), and might result in a different hardware decision. Nonetheless, the decision we made two years ago has proved to be quite effective in achieving both the specific objectives of our stated criteria as well as the larger educational objectives of integrating microcomputers into the full range of undergraduate education. The applications outlined below illustrate in different ways the importance of all these criteria.

LINKAGES AMONG COMPUTERS

The major link between the H89s and the PDP 11/70 is provided by a "terminal" program that enables the H89 to pass files back and forth to the PDP 11/70, and to operate as an ordinary terminal. This link makes it possible for both faculty and students to prepare files (programs, reports, term papers) on the H89 in stand-alone mode, then to send them to their accounts in the large storage disks controlled by the PDP 11/70. By emphasizing the stand-alone capability of the H89s for file, program and data entry, this time-consuming activity can be shifted away from the PDP 11/70, thus saving scarce terminal and CPU time.

In practice, communication between the H89s and the PDP 11/70 takes several forms. Some H89s are hard-wired to the 11/70 so that they are always ready to function as terminals or to pass files between the two systems. Others are never connected to the 11/70. Still others--especially those owned privately by faculty and students--are occasionally linked via phone modem. Since most microcomputers at St. Olaf are H89s, individuals can use any H89 to prepare material for the PDP 11/70 (e.g., reports destined to be printed using a UNIX text formatter, or data files for a major PDP 11/70 statistical package), then transfer this material to the PDP by a modem or by inserting a floppy disk into one of the campus H89s that is linked to the 11/70.

File passing also works the other way: programs designed to run on the H89s can be stored on the big 11/70 disks, and retrieved by students as needed for class assignments on the H89. One special storage account on the PDP disk functions as a warehouse for H89 programs; anyone who wants to run one of these programs need only use the "terminal" program to transfer it to an H89 disk. Thus faculty who are making class assignments on the H89 do not need to worry about preparing or distributing individual diskettes for the entire class: they simply put the program in the H89 warehouse on the PDP disk, and students can find it whenever they are doing their assignment.

Because of the power and flexibility of this satellite system, departments, individual faculty members, and students have a powerful incentive to purchase H89 microcomputers. Departments know that by purchasing equipment similar to that used elsewhere on campus, both software and knowledge developed in other contexts becomes available to them. Faculty who decide to buy home computers find it very convenient to be able to do work at home that will immediately transfer to either the H89 or the PDP system at the college. Even students are now purchasing H89s so they can write their programs or term papers in their dorm rooms, then process the disks on the public H89s that are linked to the timesharing system.

APPLICATIONS

The linkage of the H89 to the PDP 11/70 makes possible numerous interesting applications of microcomputers. We list below some of these as illustrations of the nature and variety of projects that are possible in this satellite microcomputer environment.

Word Processing. The PDP system is linked to a high speed line printer and to several formed-character NEC Spinwriter typewriter-type printers. Formerly, students writing term papers or faculty writing course notes or research reports used the PDP system for text entry, text processing, and output. In the new system, faculty and students do their entry directly on an H89 (independent of the PDP system) using one of the many editors or word processors available for the H89. (The most popular are Wordstar, VideoScribe, and Pie, a Heath text editor.) After the file is prepared and corrected, it is transferred to the PDP 11/70 for high quality or high speed printing. Approximately one third of all student themes and term papers are now written in this fashion.

Introductory Programming. Each semester about 10 per cent of St. Olaf students enroll in one or another of our quarter-credit courses in introductory programming for beginning instruction in Basic, Fortran or Pascal. Each of these languages is available on the H89s, so students can write and try out programs on the microcomputers without consuming time on the PDP 11/70. (At the beginning level, the differences in dialect between the H89 and 11/70 versions of these languages are insignificant.) Students save their work on their own diskettes, and when programs are ready to be turned in for grading they can be printed on the lineprinter linked to the PDP 11/70.

Symbolic Algebra. Even though in most respects the PDP 11/70 offers more powerful software than do the H89 microcomputers, one exception is in the area of symbolic manipulation of algebraic expressions. Whereas large mainframe computers support such powerful systems as MACSYMA, nothing comparable exists for the PDP family. However, the muMath software package which runs on Z80 systems offers many of the features of these larger packages: exact precision arithmetic to more than 1000 decimal places, symbolic differentiation and integration, algebraic simplifications, and matrix inversion in exact terms. Having the H89s available to supplement the PDP 11/70 brings muMath to the fingertips of every student and instructor; it opens up remarkable new options in teaching mathematics.

French CAI. Because of the total documentation (both hardware and software) provided for the H89 microcomputers, we were able to modify an H89 to display French characters on the screen in response to certain of the less-frequently used keyboard characters. This enables the French faculty and students to type directly in French, with software that codes the special characters into a standard form for storage on the PDP 11/70. When these files are processed for printing on the NEC Spinwriters, they will print in French as well.

Graphics. The resolution required to adequately display functions and surfaces encountered in standard college mathematics and sciences courses is somewhere between that offered by the so-called high resolution Apple (or Atari) graphics and the professional quality resolution of, for example, Tektronix equipment. However, the ability to address individual pixels on the 80 x 24 character screen of an H89 (or any standard terminal) yields a picture window with about 500 x 250 points. This resolution is adequate for school mathematics and science, yet it is not provided at a reasonable price by most terminal or microcomputer manufacturers.

For many years we have used Retrographics boards in our Lear-Siegler terminals to provide this resolution. A similar board, manufactured in England and marketed in the United States by Maxtek, provides the H89s with this point-addressing capability. Locally produced software for two and three dimensional plotting is available on our UNIX system for call by programs in Basic, Fortran, or C. In addition, two standard packages provide casual users with powerful software for plotting points, curves and surfaces, having them displayed on the user's terminal, and ultimately printed on one of the several inexpensive Houston Instrument Hi-Plot flat bed plotters that are linked to the PDP 11/70.

Mathematics Modelling. The graphics capability of the Maxtek board, combined with the algebraic power of muMath, makes the H89 microcomputer an ideal instrument for enhancing high school and beginning college mathematics. We are currently developing, under an NSF/NIE grant, three prototype units for supplementing mathematics instruction in those transitional courses commonly taught both in high school and college--algebra, functions, elementary statistics, and probability.

These three units use simulation (e.g., of a bouncing ball) to provide students with a source of data for conjecture, calculation, graphing, and subsequent analysis. Each simulation can be used at many different points in a typical mathematics course. For example, the successive heights of a bouncing ball can be introduced first as an exercise in graphing, then reused later in a study of exponential decay and infinite series. Other units emphasize probabilistic reasoning based on simulation of waiting times, and algorithmic reasoning based on uses of the muMath package.

Although these packages are designed for an H89 in stand-alone mode, the development of the material has been immeasurably aided by the linkage between the PDP 11/70 and the H89s: programming can take place on either system; storage of program modules, and subsequent electronic cut-and-paste can take place within UNIX where it is far more convenient; and the instructor's guides can be prepared for processing and printing within the UNIX system where more sophisticated software and higher quality printers are available.

Linkages with Other Microcomputers. The principles of linkage established between the PDP 11/70 and the H89s make possible similar transfer of programs and files with other microcomputers as well--using the PDP 11/70 as the central switchboard. Files and programs prepared on Apple, TRS-80, and Cromemco computers have been passed via wire or modem to our 11/70, and in certain cases where it was appropriate, passed along to an H89 for editing and compiling on that system. Direct transfer of disk data--reading an Apple disk on an H89, for example--is much harder. Some students worked on that this year, but were only partially successful.

Classroom Display. A universally vexing problem for educators using mini and microcomputers is the difficulty of displaying computer CRT screens in an average class. Typically, the home computers (such as Apple) designed for standard TV screens will work well, but do not have the full range of software used in a typical college instructional setting. Terminals and microcomputers linked to systems such as UNIX ordinarily fill the screen with so many characters that they become illegible when displayed on either a standard TV screen or a projection TV system.

We developed a response to this dilemma in two stages. First, we designed a video generator board that would attach to almost any terminal or microcomputer to convert the CRT signal in a computer display to the appropriate signal for a TV monitor. (This problem is not as trivial as it might appear.) This allowed us to put both text and graphics (from the Retrographics or Maxtek boards) on large classroom monitors, even though the text was hard to read in the back two rows of the class.

To solve this latter problem, students working in a class project designed a large character generator for the H89 with the Maxtek graphics board, so that output from the UNIX system, when arriving at an H89 in these classrooms, would be thrown on the CRT in large type--which then also appears on the classroom monitors.

Other Projects. The idea of linking microcomputers to the timesharing system has expedited several other projects as well:

-- A technical text editor that makes it possible to do word processing with mathematical, scientific, and foreign language text.
-- A color graphics system on a Cromemco S-100 computer that makes it possible to generate and edit color TV pictures.
-- A multiplexer for pen plotters that permits several plotters to be run off a single I/O port on the PDP 11/70.
-- Simulation of computer-controlled electronic devices for student laboratory experiments.
-- Laboratory exercises in operating systems and machine programming for beginning students of computer science.

While most of these projects could be done with a timesharing system alone, the presence of the microcomputers makes it possible to have far more student computer stations operating simultaneously. Moreover, the independence of the H89s permits projects--unthinkable in a timesharing environment--that involve tinkering with the operating system.

DOCUMENTATION

Computer systems that are locally developed or modified frequently lack adequate documentation for casual users. Even though UNIX is not locally produced, it suffers from inadequate documentation because it is so easy to modify: each installation is slightly different from each other one, so special local efforts must be made to inform novice and casual users of the system protocol. The H89s share this same dichotomy: because Heath provides such complete system documentation, it is easy to change everything. Hence there is no single, simple guide to HDOS for the computer novice.

When you put both UNIX and HDOS together you compound the chaos. Casual users have to struggle with two systems, neither of which is well documented. Small differences in protocol between the systems appear magnified when viewed from the perspective of novice users already uncertain about the role of computer operating systems. While St. Olaf has by no means solved this enormous problem, we have tried to provide documentation for the H89 microcomputers that capitalizes on the linkage between the H89s and the PDP 11/70.

The PDP 11/70 is a timesharing system with large central memory and terminals all over campus. Since this is an ideal vehicle for information retrieval, we decided to set up electronic documentation of the microcomputer system on the PDP 11/70 system. Electronic documentation also has the advantage of being easier to maintain, a significant advantage in the rapidly changing world of microcomputer systems.

The core of the system is a master catalogue of programs available for the microcomputers, each with brief annotations. Documentation for these programs, if short enough, is kept on-line on the PDP 11/70, and can be displayed quickly on any campus terminal by using a special reference program. Documentation that is too long for realistic on-line reading is made available in master sets in several campus locations, and is referred to by the PDP 11/70 microcomputer reference program.

Finally, and perhaps most important, we provide on-line a continually growing list of supplementary documentation, largely unrelated to specific programs, about using the St. Olaf microcomputer system. The topics covered in this supplementary series range from discussion of how to initialize and prepare disks for different purposes--storage, booting, system work--to instructions for changing bytes on a disk in order to restore accidentally deleted files. Supplementary documentation arises primarily in response to common questions that seem inadequately answered in other standard sources.

OBJECTIVES VS. ACHIEVEMENTS

Our aim at St. Olaf is to provide students with opportunities for learning about computer systems by creating and modifying them, as well as by using them. Because of this, a substantial part of the software and system work required to develop the linkages described above has been carried out by students working under the general supervision of faculty and computer staff members. Very little that we have done is new on any absolute scale; all these things have been done before, either in the computer industry or in research universities. Had we had unlimited resources, we could have purchased hardware and software that would have done everything we have done and much more.

Whatever merit our strategy has, it is in large part due to the necessary balance of budget constraints with academic objectives. By purchasing basic equipment without unnecessary frills, by developing rather than purchasing such software packages as graphics and networking, and by seeking such open operating systems as UNIX and HDOS, we have been able to permeate the campus with computing on a modest budget. Student and staff involvement in developing the system helps stretch the budget and helps spread computing to all corners of the campus.

Special Session

SIMULATION: REAL-WORLD APPLICATIONS AND THEIR IMPLICATIONS FOR UNDERGRADUATE COURSE CONTENT

Sponsored by: SCS
Chaired by: Nancy Roberts, Grad. School of Ed.
Lesley College

ABSTRACT

The panelists will each address two questions:

1. How is simulation used as a real world problem-solving aid?
2. What should the content of an undergraduate course on simulation include?

Their response to the first question will involve a description of a project with which each has been involved where simulation has been used as an aid in problem solving. These applications include simulations of chemical processes, educational simulations to aid the Ministry of Education in Mexico with decision-making, the use of simulation in an industrial setting.

In response to the second question, each panelist will again discuss their experiences teaching courses in both continuous and discrete simulation. Such topics as the following will be covered: what sort of situations are candidates for the use of simulation; what is the spectrum of computer languages available for simulation; how much emphasis should be placed on programming; what sort of statistics and probability training if needed; what place have the topics of verification and validation; and what are reasonable problems to assign?

Participants

Richard Bronson
Fairleigh Dickinson University

Ralph Deal
Kalamazoo College

Eduardo Riva, Mexico
Ministry of Education

Charles Shub
University of Vermont

Special Session

USE OF COMPUTERS IN MATHEMATICS
HOW TO GET STARTED

Sponsored by: CIMSE
Chaired by: Sheldon P. Gordon
Suffolk County Community College

ABSTRACT

This session will describe the development and use of computer centers and computer techniques at three colleges for undergraduate mathematics instruction. At Albion College the uses of the computer in the undergraduate mathematics classroom are discussed, with emphasis on the graphics capabilities of microcomputers. Several Apple programs for use in calculus and differential equations are demonstrated. Some principles and admonitions are given to users and software authors.

At the University of Florida the emphasis is on workshops and inservice training for mathematics teachers. Elementary and secondary schools are embracing computers as instructional aids even faster than universities, in many regions. Often, the most serious limiting factor is the shortage of teachers knowledgable about computers. The University of Florida school support activities have led mutually beneficial contracts between the Departments of Mathematics, Mathematics Education, and Computer Science. A variety of support programs have been developed, including summer workshops, weekend short courses, driveout meetings, and off campus courses. Some of the successes and failures will be discussed and a working set of "consultant's tools" will be described.

At Dekalb Community College the emphasis is on the development of a computer facility to implement on NSF CAUSE grant as a Math learning center. The focus covers the development of needs in the hardware and software areas as well as the process of implementing the plan.

Participants

Raymond N. Greenwell
Albion College

Mark P. Hale
University of Florida

Billie Ann Rice
Dekalb Community College

Project Presentation

THE COMPUTER AS A TOOL IN UNDERGRADUATE CURRICULUM AND BEYOND

Chaired by: Harry Hedges
Michigan State University
Charles W. Basham
Don M. M. Booker
A. Rees Clark
Elizabeth N. Derbyshire
Thomas Flynn
Susan J. Foster
Melanie Wolf Greenberg
Gail S. Lee
William Paisley

ABSTRACT

A Computer Information, Analysis, and Simulation System for Teaching in Agriculture

Charles W. Basham, Gail S. Lee, Elizabeth N. Derbyshire; College of Agriculture Sciences, Colorado State University, Ft. Collins, CO 80523

The agricultural sciences are concerned primarily with the management of biological systems to yield food, fiber, and more recently, biomass for conversion to fuel energy. Agricultural practices are based on hugh amounts of empirical data, observations, and accumulated experience. The availability of electronic digital computers provides the capability to organize this mass of information and make it available for use in decision making on a scale not previously possible.

The traditional intensive education in the biological sciences will continue to be essential for the young scientists who will be responsible for the research, development, and implementation efforts leading to solution of these problems. To meet these challenges, undergraduates in agriculture sciences should have additional training in such areas as mathematics, statistics, and computer science to enhance analytic capabilities and prepare them to work with professionals from those fields and others in a systems approach to large-scale interrelated societal problems involving agriculture. Conversely, students in other fields should have an opportunity as undergraduates to explore the application of their disciplines to agricultural problems.

This proposal centers on making the power of the computer an integral part of the teaching-learning environment. In the October 22, 1979, issue of the "Chronicle of Higher Education", F.J. Rutherford of the National Science Foundation is quoted as saying: computers "are not commonly used in the instruction of most students," and "we do not yet know how to integrate these technologies into the classroom or into the lives of students in any way that is effective and satisfying." To help "integrate these technologies into the classroom," we propose to develop a system that is convenient and effective for both instructor and student who are not expert in computer science.

This proposal encompasses three broad related objectives:

I. To develop a unified system of software linked to data files which will allow students to examine data, revise it, do operations on it, make decisions and test those decisions in a simulation.

II. To develop a set of case studies and accompanying student problems utilizing the capabilities of the software systems; these cases will be of graded complexity for each level undergraduate instruction and will include biological, economic, and societal considerations in an agricultural systems text.

III. To develop an integrated sequence of courses to prepare students to use a systems analysis approach to problems in their technical fields and broader interrelated problems.

ABSTRACT

Integrating Data Flow Models and System Dynamics in Information Systems Design Courses

Don M. M. Booker, Department of Computing and Information Systems, Pace University, 150 Nassau Street, New York, NY 10038

A large body of recent work in information system design models has encouraged the use of structured data flow models (Yourdon and Constantine, 1978; Gane and Sarson, 1977; Jackson, 1975). The computing and information systems curriculum at Pace University has emphasized incorporating the data flow approach to systems analysis and design in the two term introductory management information system design course, the on-line system design course, the data base design course, and to a lessor extent the decision support system design course, for the past several years.

The data flow model was originally conceived to achieve the goals common to other structured programming methodologies - primarily the eventual production of correct (e.g. meeting defined requirements), complete and error free program code and operating procedures constrained to exhibit only the three primary control structures of sequence, iteration and choice, and organized by heirarchical invocation.

While the data flow model has served well in teaching information system design at a detail or program design level, and as a method for requirements definition and analysis, several weaknesses have emerged. It is difficult to relate to the wider functional requirements and behavior of the business firm and its environment. It is difficult to incorporate the less well defined, non-computerized, information processes and decisions, both on an operational level and at a policy level. It is particularly difficult to integrate policy level decision support systems design methods with lower level operational information systems methods and models at a quantitative level.

While the structured data flow models are intended to be 'dynamic' models of information flow, they do not in fact lend themselves easily to quantitative estimates and analysis of actual dynamic data flows. This becomes evident at a fairly basic design level when the estimation of physical design parameters such as data volume and frequency is attempted rather than logical description and decomposition into data elements and structures. In more advanced courses such as on-line systems design and distributed systems design, this weakness becomes more apparent. It becomes extremely difficult to effect a smooth instructional transition from structured data flow design models and methods to quantitative techniques needed for performance monitoring and measurement.

In an effort to overcome these weaknesses, we have recently begun to introduce some system dynamics models and techniques (Forrester, 1961; Goodman, 1972). These models are introduced as a natural extension of the structured data flow models we have been using. This allows us to introduce a more quantitative approach and a dynamic simulation capability to the design process. The integration of detail level information system design models and wider models of the firm is facilitated. An earlier and more detailed model and analysis of the physical system is possible. Simulation of on-line and data base performance models can be included along with case studies and smaller problems.

Our instructional approach emphasizes the use of business cases for information system design rather than smaller problems. When the micro version of Dynamo is released we plan to complement the individual design cases assigned to students with more extensive interactive simulations.

References:

1. Forrester, J.W. Industrial Dynamics. Cambridge, Mass., M.I.T. Press, 1961.
2. Gane, Cris and Sarson, Trish. Structured Systems Analysis: Tools and Techniques. New York, Improved Systems Technologies, 1977.
3. Goodman, G.M. Intro to System Dynamics. Boston, Mass., Alyn and Bacon, 1972.
4. Jackson, M. Principles of Program Design. New York, Academic Press, 1975.
5. Yourdon, E. and Constantine, L. Structured Design. N.Y., Yourdon Press, 1974.

ABSTRACT

Word Processing's Usefulness for Writing Instruction

Thomas Flynn, Assistant Professor, Ohio University-Belmont, St. Clairsville, Ohio 43950

Purpose: This presentation will describe the integration of word processing (WP) into the instruction of writing at the basic and introductory composition level.

Scope: I will discuss:

1. The hard and software used
2. The amount of time needed to familiarize students with the techniques necessary for WP
3. Modifications necessary for adapting published WP instructions to classroom use
4. The disadvantages and advantages that WP offers for writing instruction.

Details:

1. Hardware: Six 16KTRS-80's, two 32KTRS-80's with expansion interfaces, two printers. Software: Radio Shack's Scripset.
2. The basic commands (saving, loading, inserting, and deleting) can be mastered in approximately four hours. More complex commands (labeling blocks and formatting) require an additional four hours.
3. Scripset comes with a complete set of instructional cassetts: a method suitable for self-instruction but inappropriate and cumbersome for classroom use. In general the instructions had to be abbreviated; however, some areas that frequently gave students trouble

such as saving to a tape had to be supplemented.

4. Disadvantages: These chiefly arise out of the limitations of the hardware. For most novices saving by tape is a time-consuming, perilous and frustrating activity. The CRT's can display only fourteen lines of text at a time, which impairs a writer's ability to match what he is about to say against what he has said. A 16K memory can only hold approximately 500 words. Although the initial text can be saved and further material put in, this method is cumbersome.

   Advantages: Despite the difficulties involved, WP offers important benefits for writing instruction. The greatest of these is that it frees the writer from preoccupation with the minutiae of writing (the mechanics of grammar, punctuation, and spelling); all of which can easily be cleared up in later revisions. The effort involved in making a hard copy and the desire to be neat impel writers to overlook their misstatements and to reject the temptation to improve their phrasing. The second advantage is that because of the facility with which WP can move words, phrases, and paragraphs around, this technique provides writers with a clearer sense of how a text can be put together. It also gives them a greater freedom to experiment with various ways to express themselves.

ABSTRACT

Use of Video Tape to Present an Introduction to Computing

Susan J. Foster, Alex Rees Clark, PhD,
Temple University Computer Activity, Broad
and Montgomery St., Philadelphia, PA 19122

The labor intensiveness of providing user training can be reduced by video tape presentation of topical training materials and general introductory subjects. This paper will describe a video taped general introduction to computing hardware and software.

People encounter computers in several ways. They may take a programming course or use computers in their jobs. Such introductions do not provide users with an overall view of how computers function and how systems are organized. Later, a user's changing needs may create a heavy load on consulting services. This consulting load can be reduced by providing appropriate user training.

At Temple, a comprehensive series of user training seminars is provided each semester. Among these for several semesters has been a one-session, two-hour seminar entitled "Everything You Always Wanted to Know About Computers, but Were Afraid to Ask". This topic's breadth required that it be taught by a relatively senior member of the consulting staff. Each two-hour session required approximately one hour to prepare. The seminar was offered approximately six times per semester, and that number was growing steadily, pushing the cost per semester into the thousands of dollars.

The video tape has several advantages over the lecture format. The standardized seminar can be presented by a junior consultant. Moreover, the tape can be made available through a media center for individual or group showing without a consultant/presenter. In the case of our tape which we call "Computers and You," our lecture presentation which ran two hours was compressed to approximately twenty-five minutes of video tape. The video tape is now presented in a one-hour period with a hand-out that contains a glossary and some amplification of points made in the tape.

Several lessons were learned in the preparation of the tape. First, one does not directly transfer an oral presentation to a video tape without making substantial changes in the organization of the information and the visual aides. Video allows for the juxtaposition of various verbal and visual information. It's possible, to show a real computer system in operation or to simulate sound. Second, the primary focus in the design of a video presentation must be on the script, which must be completely prepared in advance of any taping.

"Computers and You" was completed in January, 1982 and used in approximately a half-dozen presentations so we have not had much opportunity to evaluate its effect. We estimate that we have reduced the cost of presending our seminar by approximately two-thirds. In addition, comments by those attending the session have been generally favorable. Persons attending have taken copious notes, and the level of questions being asked of the presenters has been higher than in the lecture mode. On the basis of this initial favorable response, we intend to convert several other training sessions to video tape. Initially, we will convert those sessions that involve training in the use of statistical packages and our courses on using the operating system on our principle academic computer.

ABSTRACT

Social Research Simulation on a Microcomputer

William Paisley, Associate Professor of
Communication, Stanford University, Stanford, CA 94305

In just two decades, the use of computers to teach social research methods has progressed through five stages:

1. The use of the reproducer (wired-board card punch) and counter-sorter to recode and tabulate cases.
2. The use of computers in the IBM 7090

generation to analyze data via user-written programs. Users shared programs, but there were no widely available statistical packages.
3. The use of early MBD and SPSS packages to perform more powerful analyses, still in punched-card batch mode.
4. The use of a much larger array of statistical packages from remote job entry terminals.
5. The use of statistical packages on minicomputers and, recently, on microcomputers.

Each later stage of computer support for social research methods instruction has increased student's opportunities to manipulate the raw data -- recoding, transforming, combining -- and to compare the different results of alternative analysis procedures. However, since the statistical packages have been designed for *doing* social research rather than *teaching* social research, instructors have been forced to design "overlay procedures" to guide students through learning experiences with the statistical packages.

Also, it should be noted that, with minor exceptions such as the sampling algorithm in SPSS, the computer support available for social research methods instruction encompasses data analysis only. This is only one of seven phases of social research that courses and textbooks conventionally cover: (1) study design, (2) instrument construction, (3) sampling, (4) data gathering, (5) data editing and coding, (6) data analysis and interpretation, and (7) report preparation. Courses and textbooks seek to stimulate learning at five levels of Gagne's hierarchy (1970): (1) association (definition and use of terms), (2) discrimination of types and cases, (3) concept learning (induction), (4) rule learning (deduction), and (5) problem solving (application of induction and deduction). Few courses and textbooks succeed in teaching the fifth level.

At Stanford, undergraduate and graduate classes in social research methods are now using microcomputer programs that simulate not only the data analysis phase but also instrument construction, sampling, and data gathering. The set of programs focuses on a hypothetical California community, El Senario, whose 999 adults are extensively described in the programs' datasets. Demographic, attitudinal, and behavioral data for each person are drawn entirely from questions asked in the NORC General Social Survey (1972-1976), with percentages and measures of central tendency and dispersion that are adapted for a community composed of 40% Anglo, 30% Hispanic, 20% Black, and 10% Asian residents. In addition, kinship ties are coded across three generations of adults and social network (sociogram) choices are indicated.

El Senario is a "small world" in which students may test the effects of different question wordings, sampling fractions and proportions across strata, parametric versus nonparametric analysis procedures, etc. For example, in successive samples of m cases drawn from the population of 999 (m greater than 49 and less than 501), students can see the differing percentages or measures of central tendency. For another example, in the case of a skewed variable, they can see how successive monotonic transformations change the variable's relationship with other variables.

Whereas statistical packages such as BMD, Minitab, SAS, SPSS, etc., permit the user to repeat an analysis with changed variables or values, the default option is a single pass. Successive passes require, at a minimum, additional procedure statements. In El Senario the default is repetition with variation; the user is prompted to choose new variables, values, or other conditions for re-analysis.

El Senario is intended to increase students' skill in problem solving by showing a wide range of answers to "what if" questions, similar to the questions that a researcher must ask in field settings where the answers are costly to obtain.

At present, El Senario is programmed in Extended Basic for the TRS-80 Model II computer with 48 kilobytes of memory and at least one disk drive.

ABSTRACT

Development of a Computer Assisted Tutorial in the Use of SPSS for Application to Health Service Agencies

Melanie Wolf Greenberg, M.P.H., Texas Research Institute of Mental Sciences, 1300 Moursund Ave., Houston, Texas 77030

A tutorial designed to enable students to learn about and how to use the data definition commands in SPSS was developed at the University of Texas Health Science Center and the University of Texas School of Public Health. This tutorial is one part of a larger project, written in MULTITUTOR, designed to make proficiency in the application of SPSS available to interested health service employees.

The tutorial was structured into four main sections; introduction, examples and problems, specific data definition commands and evaluation. Students were able to direct their learning through the examples and examine the commands incorrectly specified in the "mock" SPSS sessions. The introduction section introduced the student to the lesson; its objectives and structure. The examples and problems section contained an example complete with a problem statement, a data dictionary (codebook) and an SPSS program describing the data in the dictionary. The examples were developed to serve as models for students to use while working on data definition problems. A problem was then developed for each one of the six examples. The problems began with a problem statement, a data dictionary and an opportunity for the student to enter the data definition program line by line. The line by line entries of the program were then analyzed to determine if the student entered the correct command and respective tag. Students were able to leave the program at any time and re-enter where they had left. The section containing the data definition commands allowed the student to review the definition of fourteen SPSS data definition commands, and, if warranted, practice in the use of that command through examples and problems. The evaluation section was included to evaluate the strenths and weaknesses of the tutorial for future revisions and improvements.

The tutorial was made available to users of the University of Texas Health Science Center Education and Research Computer Center in Houston. Eighteen users participated in the analysis. Most users accessed the examples and problems section and completed the first problem in approximately 49 minutes. Most users indicated that they had little problem in following the computer's instructions. Overall, most users believed that there is a need for other lessons to be developed in SPSS and other software packages.

The next step in the development process would be to field test this tutorial in the health agency setting and determine the utility of tutorials in public health agency settings along with the development of other SPSS lessons coverning data analysis commands.

Computer Based Education

A CALL FOR THE STUDY OF COMPUTER GAMES

Stewart A. Denenberg
Dept. of Computer Science
SUNY, Plattsburgh
Plattsburgh NY 12901
518-564-3608

INTRODUCTION

About six years ago, I was using the PLATO system[1] to develop a CMI/CAI package in the area of computer literacy. In the mornings, I would work very hard trying to bend PLATO's TUTOR language to my needs and, as a form of self-reward, in the late afternoon I would use the powerful PLATO electronic mail system and play its incredible (then) games. One of these games was the highly complex 'empire' whose goal it is, simply put, to conquer the universe. It is a co-operative/competitive game where you join a team and work with team members to conquer planets using armies and spies as well as engage in space dogfights a là Star Wars. However, the rules and techniques are so complex (there was a separate Plato lesson just to explain the game) that I soon became impatient and just plunged into it, figuring that I'd learn what I needed as I went along. Wrong. I kept getting blown away (literally) almost immediately and this became frustrating and depressing. Fortunately, the game has a facility whereby the user may send messages to everyone else in the game or to specified players. So I entered once more, this time prefaced by the message "New player in game...Please be gentle." Almost immediately I received a response, "Need some help?" Of course, I replied, "Yes", and I was immediately put into "tractor orbit" and protected while my mentor towed me about this electronic universe explaining the most important rules as well as some very useful tricks (such as: don't use phasers, use photon torpedoes - much more powerful.) After about half an hour of valuable instruction, I told my teacher that I thought I was ready to try my own wings and thank you very much. "No problem", he replied;" ...by the way, how old are you?", he continued. "Thirty-eight", I answered. "YIKES", he responded. "What's the problem?", I puzzled. "I'm twelve", replied my mentor.

Well, I'm sure she (or he) and I learned valuable lessons from that encounter made possible through the medium of a computer game. Since the communication was stripped of its normal sterotypic cues (sex, age, race, tone of voice, physical characteristics, smell, body language), the interaction became truly egalitarian. The conversation was enhanced and humanized by one of our most advanced technological tools (and makes an interesting counter-argument to the contention that computers reduce people to (binary) numbers, degrade them and dehumanize them). I began to change my views on the worth of computer games.

About two years ago, I overcame my natural reticence and wandered into an electronic game arcade in a shopping mall. I attempted to watch the players as a visiting anthropologist might observe a foreign culture. I couldn't understand the rapt attention they were paying to these silly games. Then I played Asteroids. I began to change my mind about the power of computer games.

If computer games could have a humanizing influence, if they could be so enjoyable and engrossing, can they also be educationally worthwhile? A common criticism of computer games is that players fritter away their time acquiring knowledge and skills that can only be used for a specific game. But that criticism can be leveled against many aspects of life itself, including formal education. When we as teachers teach the history of Egypt or matrix algebra, we are, in many instances, teaching knowledge or skills that can only be used in a unique situation and are in fact not useful in our daily lives. The saving grace for both games and formal education is that we do not propose to teach content - which can be acquired better from books in a library - we are teaching problem-solving skills; we are showing the learner the process of how to go about solving a problem, and it is the acquisition of this process that is truly important in one's education. Granted, the problem-solving process is highly context-dependent so that the skills gained from inverting a matrix are not usually transferable to gathering historical evidence. What is transferable, however, is the experience (the game) of having solved some problem, any problem, and the confidence that comes from that game. Approaching a new problem with that confidence can make the difference between success and failure; it is a necessary (but not sufficient) condition for success. If, through games, we are imparting the ability to the student to confront a problem confident that it can be brought under control, then we have no need to make excuses for computer games.

I believe that computer games are educationally important and that they can be incorporated into a computer literacy pedagogy. If this is so, then it would be useful to better understand computer games and we may begin the process of understanding by developing a taxonomy. If we can categorize computer games, we have achieved a clearer understanding of their similarities and their differences.

A SUGGESTED TAXONOMY

Perhaps a sociologist or anthropologist would be the most appropriate person to study the phenomenon of computer games and then attempt to develop a taxonomy. However, in the interests of starting the discussion, we could argue that all computer games fall into one or more of four basic categories:

1) Quest
2) Mental Skill
3) Physical Skill
4) Simulation

The Quest (or Adventure) type of game is a fantasy shared by the computer and the user. Typically the user can assume a role during the play and his goal is to maximize the acquisition of some type of treasure while minimizing his effort to acquire it. At the start of the game, the user usually is given a fixed amount of capital (of some form) to be traded for other resources (such as food, water, etc., or even skills and powers such as strength speed, invisibility, etc.). A wise (and lucky) investment at the game's beginning can lead to a happy ending for the fantasy. Other characteristics of this game type are that the user plays the role of hero in the fantasy and that the game can extend quite far in real time, being played over several days or weeks if need be.

The games of mental skill are not a disjoint category for it is clear that even the most fantastic quest game requires a modicum of mental skill. However there are many games that are primarily characterized by the large amount of mental skill involved, and these seem to be the older existing board games such as chess and checkers, although they certainly also include newer ones such as Mastermind and Othello. The computer is more than an automated board, however, and acts both as a referee and an opponent in many versions of these games. It is interesting to speculate on why a human trusts a computerized backgammon game to throw truly (pseudo)random numbers when it is the computer's turn to play.

Physicial skill games are of the arcade variety - a phenomenon that is just beginning to be studied by sociologists [2]. These games require psychomotor skill usually involving a high degree of eye-hand coordination wherein the player moves a piece through space simultaneously avoiding and destroying hazards. Games of this type are usually highly competitive and agressive and involve shooting and destroying 'enemies'. Arguments continue as to whether they constitute a healthy outlet for aggression or do, in fact, stimulate and encourage aggressive behavior. In any case they seem to be the most popular type of computer game as evidenced by the fact that millions of people can be seen in hundreds of shopping malls paying to play these games.

The Simulation type of game is also a very broad category that, in a sense, includes all of the other types: every game is some sort of a simulation of some aspect of reality. What we choose to call a simulation game here is one that is presented to the player as a model of a process to be investigated and better understood, rather than as a situation to control or to master. Weizenbaum's ELIZA [3] was perhaps one of the first of these games, simulating a Rogerian psychotherapist. The appeal of ELIZA comes from testing the "intelligence" of the model itself, trying to find situations that will "trip up" the response of the model. If, as we press the model further it continues to respond reliably and reasonably, our satisfaction with the game increases. Indeed, some current microcomputer versions of ELIZA are so over-simplified and so lacking in artificial intelligence that they are no fun to play.

Other simulation models include world or environmental models, or just about any complex system where it is not generally possible to predict system outputs when the system inputs are varied. The fun in these simulations comes from being surprised by unexpected outputs: either the input produced no change or an unpredicted change to the system. Specific models of this type indluce systems ranging from a nuclear reactor to Conway's Game of Life [4]which itself can be construed as a population model.

The highest form of simulation models

are those that allow the user to specify algorithms that in turn drive the model. "Turtles" driven by a LOGO-like [5] language can allow the user a high degree of creativity in forming complex graphical designs. The original turtle was a mechanical robot under computer control through the LOGO language; the newer versions are true computer games wherein the turtle has become an image on the television screen. Radio Shack has a game, 'Dancing Demon', which is also a CRT image that can be programmed by the player. Using menu selection, the player programs(choreographs) the demon to carry out a sequence of dance steps to a tune which is also programmed as a sequence of selected notes. The pleasure is this game seems to derive largely from the high degree of control the user/player has over the game. Indeed, control may in fact be the common element that threads its way through and contributes to the enjoyment of all types of computer games. If this is so, then the ultimate computer game, the most complex and rewarding game would be the one that offers the most control and this is, of course, the game of writing your own game program. If one has nagging puritanical doubts about the educational efficacy of games, then one may, as a teacher, gently channel the energies of his students from the playing of computer games to the writing of the game programs themselves.

Obviously there exist other, better, taxonomies. Perhaps a better way to categorize computer games is to first understand them as any other art form whose role it is to reflect a part of one's self back onto one's self: we see ourselves in our games as well as in our art. If this is so, then all computer games can be assigned to the area that they best reflect. For example, the quest games mirror our fantasies and emotions; the mental skill games, the rational part of the mind; the physical games, the nervous system and the perceptual organs and the simulation games work on our ego as well as our rational faculties. The psychology and metaphor may be mixed here but it illustrates another avenue to explore in the taxonomy of computer games. The key question to consider is: Along which dimensions can computer games be classified?

CONCLUSIONS AND FINAL THOUGHTS

We have suggested that computer games are deserving of further study and ore purpose of this paper is to encourage others to begin that study by developing a workable taxonomy. Less formally we would like to start others to thinking about why computer games are so attractive, so interesting, so enjoyable. Two possible reasons were stated (although not explicitly) in the previous section: control and creativity. Like any creative activity, we can lose ourselves in it, become one with the activity, and emerge from the experience at once tired and refreshed. Whe we play the game, we control time (it seems to stop) and although we are constrained by the rules, we feel as if we are in total control of the situation (something utterly impossible in real life). These games seem to simultaneously boost and diminish our ego--is that why they are so satisfying? Computer games raise some interesting questions about ourselves; perhaps it is not yet time to call for the formation of another ACM Special Interest Group (SIGAME), but they certainly are a phenomenon whose study promises abundant insight and possibly deeper understanding of ourselves, our computers and the relation between us.

REFERENCES

1. Denenberg, S., "A Personal Evaluation of the PLATO System", SIGCUE Bulletin, April, 1978, pp. 3-10

2. Trachtman, P., "A Generation Meets Computers on the Playing Fields of Atari", Smithsonian, Sept. 1981, pp. 50-61

3. Weizenbaum, J., "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine", CACM, Jan. 1965, pp. 36-45

4. Gardiner, M., "Mathematical Games", Scientific American, Oct. 1970, pp. 120-123 and Feb. 1971, pp. 112-117

5. Papert, S., "Teaching Children Thinking", MIT : AI Lab Memo No. 247, LOGO Memo 2, Oct. 1971

DELIVERY OF COLLEGE COURSES
USING EDUCATIONAL TECHNOLOGY

Joseph L. Pokorney
Middlesex Research Center, Inc.
3413-1/2 M Street, N.W.
Washington, D.C. 20007
(202) 333-1925

At a time when there are increasing demands for more consistent quality and cost effective education in every sector of the educational community, a variety of electronics-based technologies have recently been introduced. While these technologies provide substantial opportunities to improve the delivery of educational services, we are faced with some reluctance by educators to adapt these techniques to their current mode of teaching. In order to demonstrate the use of these educational techniques, Middlesex Research Center is conducting a pilot project for the United States Navy that applies these technologies in the delivery of freshman level college courses on board Navy ships.

PROJECT BACKGROUND

For a number of years, the U.S. Navy has offered the Program for Afloat College Education (PACE). PACE provides freshman and sophomore college level courses using the traditional college instructor on board Navy ships. Through a variety of contracts with colleges and universities, these courses are provided on board ships throughout the world. In recent years, the program has come under some criticism because of its lack of uniform quality and because the program has been unable to deliver college courses on small Navy vessels, such as submarines and fast frigates.

In 1975, the Department of Defense established the Defense Activity for Non-Traditional Education Support (DANTES). DANTES' mission is to act as a catalyst in the development of non-traditional educational delivery mechanisms for all branches of the armed services. To this end, DANTES has been funding demonstration projects for satellite distribution of telecourses, computerized scoring and testing services, adult learning center activities, and a variety of other projects to improve the delivery of educational services.

Early in 1981, DANTES solicited proposals for an educational technology demonstration project to be conducted on board U.S. Navy ships. Working with the Chief of Naval Education and Training (CNET), a Request For Proposal was issued, and a contract was awarded to Middlesex Research Center and its subcontractor, George Washington University. Work began on the project in July of 1981.

PROJECT OBJECTIVES

The overall objective of the demonstration project was to show, in a practical sense, the feasibility of employing microcomputers and video to deliver college courses on board Naval ships without instructors. Because the study objective focused on a demonstration of new technologies, funding was not provided to develop new course materials. Instead, existing off-the-shelf course materials and hardware were to be used to the extent possible.

Three different technical approaches were to be demonstrated:

- computer assisted instruction
- video instruction
- computer interactive video instruction.

A single technology was to be demonstrated on board three Navy vessels, with approximately three courses being delivered on each ship. The courses were to be provided without instructors or facilitators while the ships were deployed at sea. Finally, college credits were to be awarded to those students who completed the courses and met the course requirements. The courses delivered were to be of uniform quality, have similar requirements, and offer the academic credit of an accredited college or university.

EDUCATIONAL PROGRAM DESIGN

The overall educational program design was based on a very pragmatic approach to the demonstration project. Since development efforts for new course materials were not possible, we began the project with a review of available video and computer-assisted materials. In addition, we worked with the Columbian College of Arts and Sciences within George Washington University to identify potential courses and faculty participants. After a review of the existing materials by the MRC staff and a series of meetings with George Washington University, a tentative list of courses was drawn up that could be offered to enlisted crew members.

Once the Navy had identified the participating vessels based upon the ships' scheduled in-port dates and deployment dates, we conducted a crew survey on each of the ships. The results of the three crew surveys are shown in Exhibit I. Following analysis of the crew surveys, it was determined that six courses would be offered during the project, and these would be:

- Introductory Computer Programming
- General Psychology
- American Government
- Precalculus
- College Algebra
- General Mathematics.

George Washington University then selected academic instructors to be responsible for the content and student grades for each course. The approach to course preparation was that the MRC staff suggested course materials, such as video lessons or computer-assisted lessons, that might be part of a particular course. Working with the George Washington University academician, the staff determined how these materials fit with the existing George Washington University courses.

1. Introductory Computer Programming. This course is an introductory course in computer programming, and as such its primary focus is on Basic as a programming language. Numerous video and computer-assisted lesson materials were reviewed. Since none of these packages represented a truly comprehensive introductory programming course, it was decided that some of the computer-assisted packages could be used, and others could be modified and expanded. Of the video reviewed, only two short segments were considered to be valuable in terms of the course content. The final design for the Introductory Computer Programming course included 30 lessons using computerized instruction, two video segments, a mid-term exam, a final exam, four student programming projects, and three optional lessons on the use of graphics, sound effects, and the development of games. The textbook used was Using BASIC - An Introduction to Computer Programming by Julien Hennefeld, supplemented by Apple's BASIC Tutorial Manual.

2. General Psychology. In our review of existing materials, we found that there is very little computer assisted instruction related to college psychology. However, there is one complete telecourse on introductory psychology that was developed by the Coast Community Colleges. This telecourse is offered as a complete package, including 30 1/2-hour video segments, a student workbook, and an instructor notebook. It is designed to be compatible with an existing academic textbook on psychology. MRC leased the video materials and, working with the George Washington University instructor, packaged the video to coincide with the GWU psychology course and the textbook, Introduction to Psychology, by Hilgard, Atkinson, and Atkinson, which the instructor traditionally uses in her course. The completed General Psychology course included 17 videotape segments, three learning exercises, a student project (term paper) based on a movie shown on board the ship, and three exams. In the initial version of this course, the exams were administered by an officer/proctor on board the ship because time constraints precluded the development of computerized versions of these exams.

3. American Government. MRC identified two complete college level American Government telecourses offered through Dallas Community College. A review of these materials with the George Washington University instructor revealed that the existing course, as he taught it, covered topics that were similar to some of the video segments from the Dallas Community College courses, and he used the same textbook. Therefore, MRC arranged to lease appropriate video lessons selected from the two semester courses offered by the Dallas Community College. These video lessons were repackaged to coincide with the existing American Government course as taught at George Washington University. In addition to the textbook, Government by the People, by Burns, Peltason, and Cronin, and the complementary student study guide, a programmed instruction textbook, Elementary Political Analysis by Herbert Jacob and Robert Weissberg, was used. The final American Government course included 30 video lessons, written assignments from the textbook and the programmed instruction text, and a student project outlined in the programmed instruction book.

4. Precalculus. After reviewing existing materials, it was determined that none of the video materials dealing with Precalculus met our requirements and that very few of the computer-assisted packages were appropriate. A meeting with the George Washington University instructor resulted in a decision to take the existing GWU course and develop computerized drill and practice exercises to supplement the textbook, Precalculus, by Mustafa A. Munem and James P. Yizze. In its completed form, this course consisted of 37 computerized lessons, 12 quizzes, two exams, and a final. Both the exams and the final were conducted on the computer, and the students' answers were captured in a computer file. In addition, the students completed 12 quizzes on the computer. To assist the instructor in determining grades, the students were required to turn in worksheets showing the steps they used to develop answers for the quizzes, exams, and the final.

5. College Algebra. As with the Precalculus course, we found that there was not much instructional video available for college level algebra. After meeting with the George Washington algebra instructor, we decided to use a mix of MRC-developed computerized drill and practice exercises and purchased software lessons to supplement the textbook, College Algebra, by Vivian Shaw Groza. In its completed form, this course consisted of 37 computerized lessons, including eight quizzes, two exams, and a final. Both the exams and the final were conducted on the computer, and the student's answers were captured in a computer file. In addition, the students completed the eight quizzes on the computer. To assist the instructor in determining grades, the students were required to turn in worksheets showing the steps they used to develop answers on the quizzes, exams, and the final.

6. General Mathematics. Since video is not a media used to provide independent instruction in mathematics, we once again focused on computer assisted instruction for this course. The George Washington instructor was in favor of using the computer for drill and testing, but also wanted to require pencil and paper exercises and assignments. The textbook, Mathematical Ideas by Charles D. Miller and Vern E. Heeren, was the text of choice. In its completed form, this course consisted of 16 computerized lessons that supplemented the textbook, 33 textbook assignments, plus six exams and a final. Both the exams and the final were conducted on the computer, and the students' answers were captured in a computer file. To assist the instructor in determining grades, the students were required to turn in worksheets showing the steps they used to develop answers on the exams and the final.

MICRO MENTOR™ TECHNOLOGY OVERVIEW

The Micro Mentor system developed by MRC was used to deliver the college level courses described above to the Navy crew members. The Micro Mentor system utilizes modern electronic technology and instructional design techniques to provide cost effective, self-paced courses to students. The system can be used to deliver educational materials covering a variety of topics to learners of all ages. This technical overview is organized into two major sections, System Components and System Operations.

1. System Components. The Micro Mentor system consists of several components, including off-the-shelf commercial electronic hardware, instructional material, and MRC-developed software. The major system components are described in the following paragraphs.

a. Equipment. All of the equipment components are purchased off-the-shelf from commercial suppliers or through OEM arrangements between MRC and the manufacturer. In most cases, more than one brand of equipment could be used to perform a particular function. However, MRC has selected those component brands that appear to be most satisfactory, given the current pricing and technology available.

- Microcomputer - The microcomputer component consists of an Apple II+, 48K microcomputer with two disk drives.
- Video player - Several different video cassette players or video disk players could be used with the Micro Mentor system. At the present time, the system uses a Sony 1/2" (Beta I format) video cassette recorder (Model SLO-323).
- Video monitor - A thirteen-inch color monitor made by Amdek (Model Color I) is currently used with the Micro Mentor system, although any black and white or color monitor is acceptable.
- Interface board - A vital component of the system is an interface board, which connects the microcomputer

directly to the video device. These boards often are designed for specific combinations of a microcomputer and video player. At the present time, the Micro Mentor system is using the Cavri V interface board.

- Carrel - A reinforced fiberglas carrel that houses all of the equipment is provided with the system. This carrel, which is manufactured by Coloney Productions, Inc., allows easy access to the system components while providing for secured storage of the system when it is not in use.

b. Software. Two major software components are used in the Micro Mentor system: administrative software and authoring software. They are described in the following two paragraphs.

- Administrative software - The administrative software is proprietary and was developed by MRC. It manages the sequence and rate at which students take lessons and records student answers or responses, depending upon the nature of the lesson. The administrative software also creates an audit file, which records the students' progress and annotates the way in which they proceed through the course materials.
- Authoring software - Authoring software is used by the MRC staff to develop individual lessons for various instructional packages. The authoring software that MRC currently uses -- which includes Pilot, Cavri V, and the Basic programming language -- is available through equipment suppliers or software firms.

c. Lesson Materials. Since the Micro Mentor system utilizes a wide selection of lesson materials in different media, a particular course or educational program may use any combination of these materials. The primary instructional components of the Micro Mentor system are described below.

- Video - Videotaped lesson materials have been purchased or leased from existing sources, primarily universities and colleges, that have been producing telecourses that are normally broadcast over public television. These telecourses typically consist of about 30 video segments of 30 minutes each. MRC uses selected lessons from these existing resources. In the future, MRC will most likely begin to produce video that is specifically scripted for the Micro Mentor system.
- Computer assisted instruction - Computer assisted instruction (CAI) lessons provide a variety of student exercises, drill and practice, and quiz material relating to a specific subject or course. CAI lessons that address individual topics can be purchased; but, for the most part, complete courses are not available at this time for microcomputer application. For the Micro Mentor courses, MRC is using a variety of purchased CAI lessons, as well as MRC-developed lessons.
- Textbooks - The Micro Mentor system uses commercially available textbooks. The textbooks are selected by the George Washington University course instructor.
- Notebooks - Since the Micro Mentor system is designed to be instructor-independent and individually-paced, a key component of the system is the student notebook, which is developed by MRC and is considered proprietary. The course notebook provides detailed instructions for the student and presents the course materials in a logical sequence that is easy for the student to follow. The notebook also includes guidance on individualized learning techniques and other materials appropriate to the course.

2. System Operations. When a student takes a course of study via Micro Mentor, he or she works from the student notebook; the Lesson Indexes identify the learning materials associated with each lesson.

Many of the lessons consist of a combination of textbook assignments (either reading or exercises), video lessons, and computer assisted instruction. Other assignments, such as a project, term paper, or test, are also presented as individual lessons. The student uses the Micro Mentor system to watch the assigned video segments and to perform all CAI requirements. All of this activity is recorded on the individual's administrative diskette.

Depending upon the content of the course and the materials that are available, a course may consist of 30 video lessons, 40 to 50 computer assisted instruction lessons, or some combination of these components. Two to six computer assisted lessons may be packaged on one diskette. Each video lesson is recorded separately on a 1/2" Beta format tape. A single set of lesson materials is provided for each course.

In a more sophisticated course design, which MRC expects to use for new Micro Mentor courses, a much more interactive presentation will be developed so that the microcomputer software directly controls the video segments that the student is watching. In this model, we envision shorter but highly content-oriented video segments, followed by CAI lessons that will provide immediate reinforcment through specific questions presented via the microcomputer. Based upon the student's responses, alternate video segments will be called up by the computer to meet the student's reinforcement requirements or to modify the level of material presented and adjust the student's pace.

Quizzes, tests, and examinations may be administered either wholly or in part by the microcomputer. For math courses or other course material that is easily quantified and for which answers can easily be entered into the computer, quizzes and other tests can be self-administered. The student's answers can be recorded in computer files that are not accessible to the student, and a reasonable level of test security can be maintained. For courses requiring narrative answers or timed examinations, the computer can be used to present the questions and to time the student activity. A certain amount of proctoring is required for some examinations, since written responses or supporting worksheets must be collected and secured. At the conclusion of a course, the student retains the notebook and textbook(s); the lesson diskettes, administrative diskettes, and video materials are returned to MRC or to the person who is coordinating the training delivery.

ACADEMIC ACCREDITATION

Since the Micro Mentor system was used to deliver college courses in the Navy project, a university affiliation was essential. Association with an accredited university was needed for the course content review, evaluation of student results, assignment of grades, and maintenance of student transcripts. George Washington University in Washington D.C. is providing this affiliation for MRC at this time. Any university that is administratively flexible and accepting of alternative learning environments could be involved in the delivery of college courses through the Micro Mentor system.

INITIAL RESULTS

The first ship (a nuclear submarine) to use the Micro Mentor delivery system left port on October 13, 1981, and returned 71 days later on December 22. When the submarine departed, 37 enlisted men were enrolled in one of the three courses offered on the system. The three courses on this particular ship were General Psychology, Introductory Computer Programming, and Precalculus. Based on the evaluation instrument that was completed by the participants prior to the submarine's return to port in December, the project was exceptionally well received by the students. A summary of the evaluation results reflecting participant attitudes toward the Micro Mentor delivery system is shown in Exhibit II. As can be seen by the exhibit, the students enthusiastically endorsed the Micro Mentor delivery mechanism, found it reasonably easy to use, and indicated that the system and the courses presented generally met or exceeded their expectations.

As a result of severe constraints on off-duty time and pressures for individuals to participate in job-related training, only 22 students completed the courses as scheduled, for an overall completion rate of 60%. The individual completion rate varied substantially among the courses, with 72% of the enrollees completing the Introductory Computer Programming course.

The final student grades for completed courses are one indication of the overall effectiveness of the educational technology delivery system. Although the grades for individual courses do not provide sufficient information to support much analysis, the overall grades for the 22 students are significant. The grades that were received by the students on the first project ship appear to be distributed approximately the same as grades achieved through traditional classroom instruction. The grades for the initial students are tabulated in the table below.

INITIAL PARTICIPANTS' GRADES

Course	A's	B's	C's	D's	F's
Computer Programming	3	4	3	1	0
Precalculus	1	2	0	1	1
General Psychology	0	5	1	0	0
TOTAL	4	11	4	2	1

PRELIMINARY CONCLUSION

It is unreasonable to make or to even suggest conclusive results based on one ship's success and a small sample size of 37 participants. However, an additional ship left port during January and will return in June, 1982. This ship, a fast frigate, has 37 students enrolled in one of the three courses offered. We expect the completion ratio for this second ship to be substantially the same as the submarine crew, which will give us between 50-75 student results to evaluate at the end of the project. In addition, because the operational environment on board the fast frigate is substantially different from the submarine, we expect the results to indicate the feasibility of using the Micro Mentor system on board most of the Navy's smaller vessels.

The success of this project will, we believe, validate the use of this new technology to provide instructor independent college level courses as an alternative to conventional modes of higher education delivery. The completion rate is comparable to the rate obtained when onboard instructors are used. The grades are average to above average, and the students who completed their courses all rated the delivery methods as highly successful and would like to take more courses. In addition, the Micro Mentor approach offers uniform course quality throughout a variety of non-traditional instructional settings. These settings might include shipboard locations, remote military installations, libraries, job sites, and individual residences. Once we refine these prototype courses, develop additional courses, and incorporate the sophisticated interactive video aspects, we will have a system that answers the call for educational quality and cost effectiveness. By doing this demonstration project and sharing our findings, it is our hope that educators will see the benefits of technology and understand the need for their support and involvement in similar projects.

EXHIBIT I

CREW SURVEY RESULTS

Courses	Number of Requests for Each Course SHIP A	SHIP B	SHIP C
Computer Programming	17	14	13
Precalculus	16	0	0
College Algebra	16	0	10
Psychology	11	5	6
Nat. & Phy. Sci.	10	2	2
Literature	8	1	0
Mathematics	7	10	0
American His. 1877+	6	4	6
Biology	5	1	1
Composition	4	6	0
American Gov. I	4	0	0
American Gov.	4	5	8
American His. pre 1877	3	3	0
Env. Studies	3	1	0
Hum. & Fine Arts	3	2	1
Earth Science	2	3	2
Sociology	1	5	3
Other	1	1	2

EXHIBIT II

SHIP A's EVALUATION RESULTS

100%	SAID COURSES MET OR EXCEEDED THEIR EXPECTATIONS
90%	RATED COURSES AS EXCELLENT OR GOOD
100%	SAID THEY WOULD TAKE MORE COURSES
95%	SAID COURSE CONTENT WAS CLEAR AND TIMELY
95%	SAID LESSON ASSIGNMENTS WERE CLEAR
60%	COMPLETED THE COURSES

BAFFLES: A New Deductive Game

James D. Spain
Department of Biological Sciences
Michigan Technological University
Houghton, Michigan 49931
(906) 487-2029

One striking result of the computer revolution has been the great burst of interest and involvement of young people in computer games of one sort or another. Clearly, a significant amount of an average youth's time is spent interacting with computerized games. There is little doubt that many have become highly skilled at performing certain specialized tasks, most of which require a high degree of eye-hand coordination for success. Thus, there is little question that computer games are capable of teaching certain specialized behavior patterns. Clearly, we could be teaching many skill-related jobs by gaming-type simulations. Great potential would appear to exist for training air traffic controllers, vehicle operators, and defense-related professionals. Such clear applications of computer games suggest that there is also potential for teaching higher level skills which depend more on logical-deductive reasoning than those previously mentioned. The objective of this paper is to explore one game that appears to offer such a potential.

Considerable effort has been expended in studying the potential of games in education. When computers first became available it was logical to try using computerized games for instruction. The results were for the most part rather disappointing, so that after an initial flurry of activity, interest appeared to subside. In general, most instructors consider that computer games provide relatively little educational value compared to more structured educational programs. Recently, however, there has been a rebirth of interest in educational games. McVay (1980) explored the important attributes of educational games in order to provide a starting point for developing better computer-aided instruction. Some of the attractive characteristics of educational games include: high level of motivation, clear and consistant goals, strong player interaction, simplicity, and creativity. Neideffer and Evans (1981) employed principal components analysis to empirically determine what makes games interesting to play. These authors pointed out that interest is a key factor, as an uninteresting game will receive little play and thus cannot be expected to teach much. Malone (1981), in a similar study, concluded that key features of any instructional game must be challange, fantasy and curiosity. The motivation that stems from interest is, of course, the key feature of games that make them attractive from an educational standpoint.

A major problem with any educational game is to clearly define what concepts or skills the game actually teaches. This subject was addressed by Moshell, Amann, and Baird (1980) who attempted to design an instructional game with a particular set of educational objectives in mind. However, this is a difficult task, and it is likely that many games have evolved without defined educational objectives. Most start with what might be called a focus of interest and gradually add the rules and structure of the full-blown game. Their educational value, if any, develops in a somewhat fortuitous fashion. Thus, many games clearly train the player to make logical-deductive decisions, although the real world application of the specific type of decisions involved may not be altogether clear. Chess is somewhat of an exception in that it has been recognized for centuries as playing a significant role in training individuals for positions of power.

We became interested in the potential for instructional games at Michigan Tech when a game called BAFFLES was written as a component of the SUMIT courseware development project (Spain, 1981). BAFFLES is similar to a board game called Black Box - The Ultimate Game of Hide and Seek (copyright 1978 by Parker Brothers, Division of General Mills Fun Group, Inc.). In Black Box, a player (seeker) attempts to determine 4 or 5 hidden locations which were defined by a second player (hider). The seeker gains clues about the hidden locations by sending rays into the 8 by 8 field that defines the box. The hider then reports to the seeker whether the ray hits a location, reflects back, bends off, or misses altogether. Because of the deductive logic involved, the Black Box game has great potential for training students to employ deductive reasoning. The chief disadvantage of the game is that it requires a second player with a clear understanding of the rules so that clues are accurately presented to the seeker. BAFFLES circumvents this problem by having the computer keep track of the rules of play and present the results of the player's decisions.

In the BAFFLES game, the player is presented with a box containing a known number of reflective baffles having specific locations and orientations. During play, the box appears to be closed so that the player must deduce the locations of the baffles by directing laser beams into the box from various locations around the sides. The baffles are all oriented at 45 degrees to the entering beams so that beams are always deflected by 90 degrees when they hit a baffle. If no baffles are hit by a given beam, the beam simply passes through and exits on the opposite side of the box. However, a beam may hit several baffles and exit from the box at an unpredictable location. Figure 1 shows how laser beams are deflected by baffles.

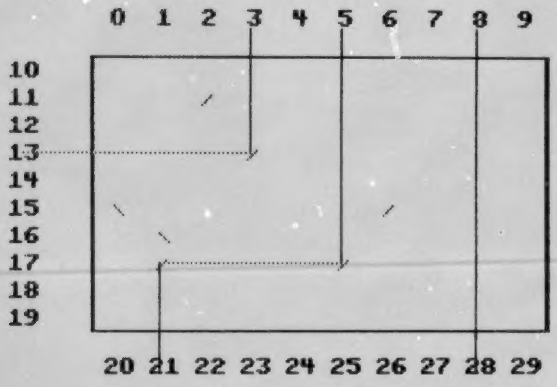


Figure 1. Laser beam trajectories in an "open" baffle box.

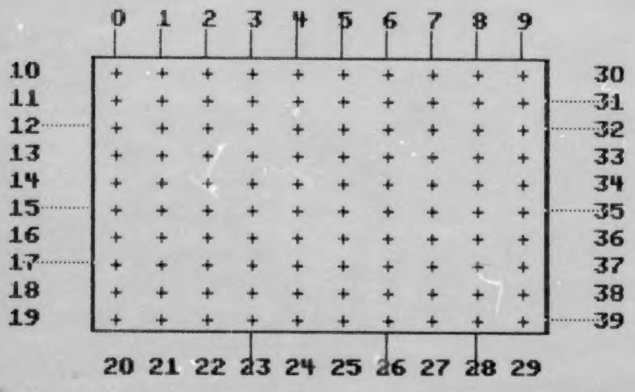


Figure 2. "Closed" baffle box with potential baffle locations as seen by a player during the course of a game.

The objective of the game is to determine the location and orientation of all the baffles by analyzing the reflections of the laser beams. The time involved in reaching the solution is not important. The final score is based only on the number of correctly identified baffles and the number of beams used.

When the player is satisfied that he or she knows the location and orientation of each baffle, the player enters the coordinates of each and its tilt into the computer. After the locations are checked by the computer and the score is calculated, the box is displayed in the open condition so that the baffles and the beam paths are fully exposed.

When a player is first introduced to the game, he or she is given an opportunity to go through the game once with the baffles exposed and the laser beam reflections shown. This gives the player a chance to develop a mental picture of the reflective nature of the baffles and the method of directing beams into the box. The location of the baffles is determined by random numbers so that a different pattern is produced for each new game. The player is given an opportunity to select the number of baffles to be placed and thus vary the level of difficulty. The more baffles involved, the more multiple reflections, and the more challenging it is to determine the locations of all the baffles.

Each beam is initiated by having the player select a number to define a beam entry location and enter it into the computer. There are 40 entry locations spaced around the sides of the box. These are designated 0 to 9 along the top, 10 to 19 down the left side, 20 to 29 along the bottom, and 30 to 39 going down the right side. After the beam entry location is entered into the computer, a colored line representing the laser beam is drawn from the entry location into the box. Then, the computer determines the pathway within the box without disclosing it to the player. The line picks up again after it exits from the box at one of the other 39 locations. Because all baffles are set at 45 degrees, a beam cannot be reflected straight back. Any reflections will always be to the right or left of the initial trajectory. It can be shown that beams always follow fully reversible pathways. That is, if a beam enters at 10 and exits at 22, then a beam entering at 22 will always exit at 10. Therefore, all potential pathways in the baffle box may be completely explored by 20 appropriately selected laser beams.

In order to facilitate the analysis, the laser beams are color coded based on the side of the box from which they originate. After the beam trajectory is drawn, the player is given a short message identifying the input and output locations in case the output line was not observed. In addition, a record of input and output locations for each beam is retained in memory so that the player may review the results at any time during the course of the game. The player may also wish to keep a written record to save time. The 100 potential baffle locations are marked on the surface of the box to facilitate analysis.

As the game proceeds, and the locations of baffles become evident, it is useful to mark their hypothetical location on the glass of the video screen using a wax pencil or crayon. This allows the player to check the consistancy of all the observed reflections. After all the evidence is accounted for in this way, the player ends the game and begins entering the location and orientation of each baffle.

Baffle locations are specified by their x,y coordinates based on the numbers along the top (x values) and left side (y values). Thus, the top baffle in Figure 1 is located at 2,11 and it is tilted (oriented) to the right. The beam from source 3 strikes a baffle at 3,13 that has a tilt to the right. After the location and orientation of each baffle is entered into the computer, the score is calculated based on the percent of correct identifications and the number of beams employed. The computer then displays the inside of the box with the actual locations of baffles so that the player can see where errors, if any, were made.

The game demonstrates the following attributes associated with the scientific method:

1. Characteristics of the system (location of baffles) may be hypothesized from observed system behavior (outputs from given inputs).
2. Testable predictions can be made based upon hypothesized characteristics (baffles at specific locations should cause predictable reflections).
3. Predictions may be tested and hypotheses confirmed or rejected as a result of performing experiments (laser beams from another source may be directed at a suspected location to confirm the presence of a baffle).
4. Total system behavior must be consistant with multiple observations (multiple reflections).
5. A researcher must announce his or her perception of the system (enter baffle locations) before the true system structure is disclosed.
6. Success is based upon the correctness of the final results presented, not on how fast the solution was attained.
7. Sometimes one is unable to distinguish between alternate hypotheses with available data. This may initially seem to be a flaw in the game. However, this is a situation which is often encountered in scientific research.

Thus, it is believed that the BAFFLES game provides effective training in the use of the scientific method. A player usually develops a strategy of directing beams in some orderly fashion until a deflection is observed. Then he or she uses the direction of bending to predict the orientation of the baffle. The experienced player will typically test the prediction by directing a second beam at the same location from a different source. Sometimes the resulting output supports the original contention and sometimes it suggests that more than a single reflection is involved. This leads to a new hypothesis about potential baffle locations and new tests are devised to confirm or reject the new hypothesis. Thus, the player proceeds from one hypothesis testing to the next until all the baffles are accounted for. The very nature of the game forces the player to adopt a strategy that is very similar to that which is employed in most research situations.

Other games have been designed to train in the use of the scientific method. An excellent example is TRIBBLES, a program written by Ruth Von Blum and Thomas Hursh (1976). This game involves a fantasy planet inhabited by small round organisms called tribbles. The objective of the game is to use the scientific method to work out the pattern of tribble reproduction and death. During the course of the game students are led to make observations, formulate testable explanations, make predictions, and observe the results of their tests. The computer generates the necessary information about tribble reproduction and death. Although TRIBBLES is recommended for use in general biology instruction, it has broad applicability for all students of science. An Apple* version of TRIBBLES is available from CONDUIT (P.O. Box 388, Iowa City, IA 52244).

BAFFLES is somewhat related to several other common computer games. Most are designed to familiarize students with the location and designation of points on a coordinate plane. BATTLE by Ray Westergard and SALVO by Lawrence Siegel are both based on the board game called Battleship. HURKLE by Bob Albrecht and MUGWMP by Bud Valenti and Bob Albrecht are two other search games that have been employed for educational purposes. All these games are described in 101 Basic Computer Games by Ahl (1974).

BAFFLES has many features which should make it fun to play according to the criteria set forth by Malone (1981). The game has a clearly defined goal of finding the baffles that have been randomly placed in two dimensional space. The scoring is based simply on the number of baffles identified and the number of beams used. The game employs some nice graphic and audio effects. There is action in the form of colored beams drawn on the high resolution screen. The player is called upon to fantasize a bit in visualizing the trajectories of the laser beams. Curiosity builds as the player completes the game and seeks to find out the actual baffle pattern. The game can be quite challenging when 10 or more baffles are involved. On the other hand, most players should have little difficulty finding all the baffles when only 5 are employed. All players should be able to achieve a certain measure of satisfaction from the game. Thus, by several criteria, BAFFLES should provide more than average motivation to players.

The BAFFLES program is written in Applesoft* Basic and is implemented on the 48k Apple II Plus* microcomputer. The program employs both HGR and HGR2 to draw the baffle box in both the closed and open configuration. Laser beam trajectories are drawn on both pages of high resolution graphics simultaneously, so that at the end of the game, it is a simple matter to switch to HGR page 2 in order to display the baffles and the beam reflections which they cause. Baffle locations are also retained as elements of a 10 by 10 matrix. Blank locations are designated by zeros and a baffle is designated by either 1 or 2 depending upon orientation, either right or left. Thus, a simulated beam moves along a column (or row) of the matrix until either a 1 or 2 is encountered. Then, depending upon whether it is 1 or 2, the simulated beam moves either right or left (up or down) along the row (or column). The output from the matrix process is used to drive the graphics display of laser beams. The program is being evaluated by CONDUIT and should be available soon on diskette from this source.

It would seem that the development of educational games is largely an evolutionary process. As a result of microcomputers, we can expect to see a large number of games being developed in the next few years. Most represent variations of a relatively few game types. Those which are really interesting can be expected to survive and flourish. Some will have more educational value than others. The role of the educator should be to assist in the selection process by seeking out those games which have significant educational potential. Often these can be slightly modified to enhance their instructional potential. Through this process, we should ultimately have a battery of computer games with clearly defined educational objectives.

Bibliography

Ahl, David H. 1974. 101 Basic Computer Games. Digital Equipment Corporation, Maynard, MA. 249 p.

Malone, Thomas W. 1981. What Makes Things Fun to Learn? A Study Intrinsically Motivating Computer Games. Pipeline 6 (2) 49-51.

McVay, P. O. 1980. Tapping the Appeal of Games in Instruction. Proceedings of the National Educational Computing Conference, 2: 271-275.

Moshell, J. W., G. W. Amann, and W. E. Baird. 1980. Structured Gaming: Play and Work in High School Computer Science. Proceedings of the National Educational Computing Conference. 266-270.

Neideffer, Jerry D., and Selby H. Evans. 1981. Games People Play: Development of a Method to Assess Interest in Instructional Games. Proceedings of the National Educational Computing Conference. 259-263.

Spain, James D. 1981. The SUMIT Courseware Development Project. Proceedings of the National Educational Computing Conference. 113.

VonBlum, R., and T. M. Hursh. 1976. Tribbles: An Introduction to the Scientific Method. Project SABLE, The University of California, Berkeley.

*Apple, Applesoft, and Apple II are all registered trademarks of the Apple Computer Corporation, Cupertino, California.

USER POWER

A Discussion and Real-Time Demonstration of Valuable Programming Features

Richard Cornelius
Department of Chemistry
Wichita State University
Wichita, KS 67208
316-689-3120

This paper will examine (and demonstrate on a large video screen) those software features that can give a user a feeling of power while operating a computer. This feeling of "User Power" provides strong motivation for individuals to use computers again and again.

Depending upon a person's viewpoint, computers may be exciting and inviting or they may be mysterious and threatening. Many times a persistent attitude is formed during an individual's first encounter with computers. Because these machines are destined to grow in the influence that they have over our daily lives, we educators have a responsibility to ensure that students' first encounters (as well as their subsequent interactions) with computers leave them with positive feelings.

USER-FRIENDLINESS OR USER POWER?

The term "user-friendliness" has been used to describe those conditions which make a computer easy to use, prevent user frustration, and generally leave the user with the feeling that a computer is being "friendly" to them. Increasing "user-friendliness" is one way to promote positive feelings of students toward computers. A different examination of what makes computers attractive, however, may serve better as an aid in writing and evaluating software for student use. This alternative approach carries the rubric "User Power." The feeling of power is a strong force, perhaps the strongest force, that draws people toward computers. Nearly any individual enjoys the ability to control a situation, to direct events, or to command activities to occur. Computers can provide all of these capabilities, and the extent to which they do so will, in large part, determine the extent to which students respond favorably to their use.

The areas in which attention should be directed in order to make the user feel powerful can be divided into five categories:

I. Documentation
II. User-Machine interaction
III. Ability of program to carry out tasks
IV. Program's use of machine
V. User interaction with the program

The first category, documentation, is perhaps the most neglected area. In order for an individual to operate a computer program, that person must first understand the capabilities of the program. Rarely can that documentation be effectively presented on the computer screen. For most programs the necessary documentation is primarily textual information that is far more efficiently presented on paper.

The user-machine interaction is an item over which a software author may not have control. Usually individuals search for programs that will run on a computer that has already been purchased (although I have suggested elsewhere that the purchase of software should actually preceed the purchase of hardware). Creative programming can overcome only a few of the design limitations of a particular computer.

Of course no program can give the user a sense of power unless the program can itself accomplish some powerful tasks. The importance of the ability of a program to carry out tasks is highly dependent upon the type of task that is to be carried out, but the more the program can do, the more powerful the user can feel.

The extent to which the program makes use of the capabilities of a particular machine may strongly affect the ability of the user to recognize the power that is available. Sound, graphics, and color are three hardware features that may be available to increase the amount of information that the user receives.

The last category, user interaction with the program, can be divided into five areas which are under the control of the software author. These five areas are identified in the form of programming guidelines, and relevant program features are indicated under each guideline.

A. Structure of input should be simple
    1. Easy to understand format
    2. Responses which fit format
    3. Limited number of keystrokes
B. Expectations should be clearly indicated
    1. Detailed instructions on screen at time of input
    2. Indication of nature and magnitude of entry
    3. Identification of default options
C. User should be protected from inadvertent action
    1. Absence of program aborts

    2. Rejection of invalid entries
    3. Opportunities for reconsideration
- D. Program should make informative responses
    1. Response within 1/10 second
    2. Identification of current activity
- E. User should have control of program flow
    1. Editing features
    2. Multiple paths of action
    3. Opportunity to begin again

All of the features identified above will be demonstrated in real time on a large screen projection television display.

Writing software which gives the user a feeling of power requires attention to the smallest details on the part of the software author. Planning for all of the desired features from the fundamental design stage is essential, and programming expertise is necessary as well. Programs which fulfill the five guidelines above also take more time to write, but the time is worth it if the product is a program which makes people come back to use it again and again.

A MICROCOMPUTER LAB:
MIXING APPLES AND "ORANGES"

by
Tom Dooley, Chairperson
and
Randy Truax,
Microcomputer Lab Assistant
Computer Information Systems
Aquinas College
Grand Rapids, Michigan
616-459-8281

ABSTRACT

1) PETs, Apples, TRS-80s, etc. can now talk to one common powerful microcomputer, the OSI-C3 (from Ohio Scientific Instruments); 2) Because of the multiprogramming potential of the C3 system and the 74 million-byte disk, the networking and distributed processing potential is extensive; 3) in this article is a description of one model for a microcomputer lab, 4) Two appendices summarize the capabilities of both the network and the OSI-C3.

I. INTRODUCTION

Can you remember the heady and exciting days of the mid to late 1970s when the advent of microcomputers turned our educational world around? I vividly recall how I raced across the State to see a microcomputer which was described as "the first computer off an assembly line". It was being marketed for $795 and had the compute power which only months earlier might have cost 10 to 20 times that.

Soon other vendors had their own microcomputers on the market, many authors were trying to help sift through the devices on the market and develop criteria to assist potential users in making intelligent choices (Dooley, 1979; Herstein, 1979; Thomas and McClain, 1979). The specific criteria were fairly consistent from author to author: speed, cost, type of architecture, available application software, power of system software, service, popularity, etc. While this advice provided significant help, there were also some difficult problems hiding around the corner.

Problem 1: "duplication of peripheral devices." Many schools found the $800 price tag of Commodore's Pets to be very attractive. Since Commodore began the 3-for-the-price-of-2 educational discount, many schools find the PET very attractive for introducing Basic and fundamental concepts. We, at Aquinas College, have purchased over 25 PETs and routinely have 300 students per semester using them. However, our budget can't stand 25 printers and disk drives. So to date, we have bought only two of each and students go through some inconvenience--and a bit of gymnastics to SAVE-VERIFY-LOAD-LIST a couples of times in order to get hard copy

Problem 2: "you will always want or need a different computer." I'm sure you've seen the face of a PET user glisten the first time he saw the color and graphics of the Apple; he had to have one, right? Or what about your music instructor who saw some of the packages from Minnesota Educational Computing Consortium (MECC)? And have you seen the statistics package of the TRS-80? Or maybe you just philosophically believe that students should be exposed to more than one vendor's products. The upshot is disturbingly the same: you will soon be buying another type of computer. Oh yes, the printers, disks, card scanners, plotters, etc. which worked on your first series of machines don't work on your second or third brand. And you'll have a Commodore printer next to a Paper Tiger on the Apple.

Problem 3: "the cost of multiplying the disks." There's nothing like a good introduction to Basic to get students interested in learning the file-handling capabilities of the microcomputer. This, of course, means that you will be adding disk drives to multiple microcomputers or at least one type of your microcomputers. And perhaps you want one or more machines to have dual disks to facilitate the disk copying procedures. No doubt about it; you're into an expensive venture. And when you get done, your Apple disks don't work well for Pets, Ataris, and TRS-80s.

II. TOWARD A SOLUTION

The above problem list is only a beginning; I'm sure we could generate a dozen more. Each advancing step will bring some of its own problems. At Aquinas College, we have tried to keep generating solutions as the problems inevitably show themselves. We've shared our ideas with more than a dozen other schools and described our efforts to people at the World Conference of Computers in Education (WCCE-81) in Switzerland last summer. Our model has helped us today and it's also something which we believe is flexible enough to accomodate tomorrow's innovations and problems. I'll describe it in two steps: a) an integrated microcomputer network, and b) a model for a microcomputer lab.

A. A Network. There are a few interesting preliminaries: because the Pets, Apples, etc. are computers, you would expect them to be much more powerful than they at first appear. Because most of the microcomputers we have talked about so far are 6502-based, you would also expect them to have stronger family ties with each other than they presently do.

So why not have all the microcomputers communicate with each other or with a common microcomputer? The economic advantage would be enormous; think of it: one fast disk and one good printer for all of the microcomputers rather than for each microcomputer or for each vendor-type.

The logic was so compelling and the economics so attractive, we decided to write some software to allow PETs, Apples, Ataris, TRS-80s, and OSI 4Ps, 8Ps, and 1Ps to all talk with a particularly powerful but inexpensive hard-disk based system (called a "C3") from Ohio Scientific Instruments (OSI). So now students and faculty have the option of either using any microcomputer in a stand alone function or they can touch a small switch on the microcomputer and be immediately in the network mode. The power of the C3 system is then available to the microcomputer user. The user enters a password and is then allowed access to varying parts of the C3 74 million-byte hard disk. Programs can be downloaded from the C3 to the microcomputers or saved in less than a second.

The C3 system allows multiprocessing for as many concurrent users as the number of memory partitions you have (we presumably have six partitions). This, combined with the extensive resources of a 74 million-byte disk provides a very powerful system. The hard disk can be divided into many separate systems and secured with ID's and passwords; we presently have the disk structured as follows:

System 1: system software and utilities
System 2: games
System 3: CAI programs
System 4: Word processing software and files
System 5: Programs to allow faculty to create review modules for students, and data files created by faculty
System 6: A Database Management System
System 7: Student and teacher files for advanced Basic
System 8: Business systems programs

We have added many other refinements to the system:

1. a clock automatically backs up systems on a small cassette (13 million-byte) tape drive
2. a vocalizer entertains most students with its synthetic voice; a blind student will use it for communicating with the computer
3. An S-100 computer looks to the hard disk for its files
4. An entertaining color organ also uses the hard disk for its color-pattern instructions.

You can see the power and flexibility of a central microcomputer. Its potential is extensive, and the price is right. An OSI C3 doesn't begin in the $50,000-$100,000 range; you can get one for $15,000. It doesn't take people long to see that a system like a C3 is a very cost efficient friend of their microcomputers. The disk drives and printers for Apples, PETs, Ataris, Radio Shacks, etc. add up fast.

B. A model for a microcomputer lab. We are pretty demanding on our microcomputer friends; they have to earn their keep. So we pondered long and hard how we could use their power most efficiently. I'll briefly describe our present model because it has helped other schools integrate their computing power.

Our OSI C3 is presently configured with six partitions which allows us multiprogramming. One of the partitions runs the network system described above. Our question then was how to best use the other five parts.

As shown in figure 1, the C3 is connected to a "passive connector panel" and through this panel to every office, classroom, and lab in the building. As a result of this design, any microcomputer can be used in any room of the building both as an independent unit using its own power or as a member of the larger network.

The "passive connector board" (figure 2) is a home-made device which allows us to activate the communication wiring to any part of the building. Presently, we have all the microcomputers in the lab active through the "carrel" ports, which leaves five other rooms capable of being activated at any given time. The C3 can grow to 16 partitions and we built the panel for a maximum of two C3 systems.

We have teleprocessing via modems in an Apple and TRS-80 in the lab. We use Edunet to access many schools throughout the country. Our Apple is capable of simulating a Tektronix 1410 so we often use the Apple to talk to Michigan State University's computer system to access the extensive geographic data files. A plotter and digitizer are attached to the Apple.

One classroom was specially configured with a 6 foot screen and VTR circuitry to allow a class to view the CRT display simultaneously with an instruc-operating a microcomputer at the front of the room. The classrooms have communication tie to the lab and via telephone to the outside computer networks.

III. THE FUTURE

We have been convinced of the value of integrating our diverse microcomputers; the fact that it is basically accomplished has been very encouraging. We continue to add new functions, while insisting that new developments remain integral to the lab. We envision the day in the near future when one C3 handles disk transfer and one handles peripheral Input/Output while dedicated CPU's are freed to concentrate on computational tasks. This also means the CPU's could be some of the new 16-bit processors. An integrated approach is not only an interesting idea; it saves money and more fully uses the computing resources.

MICROCOMPUTER NETWORK

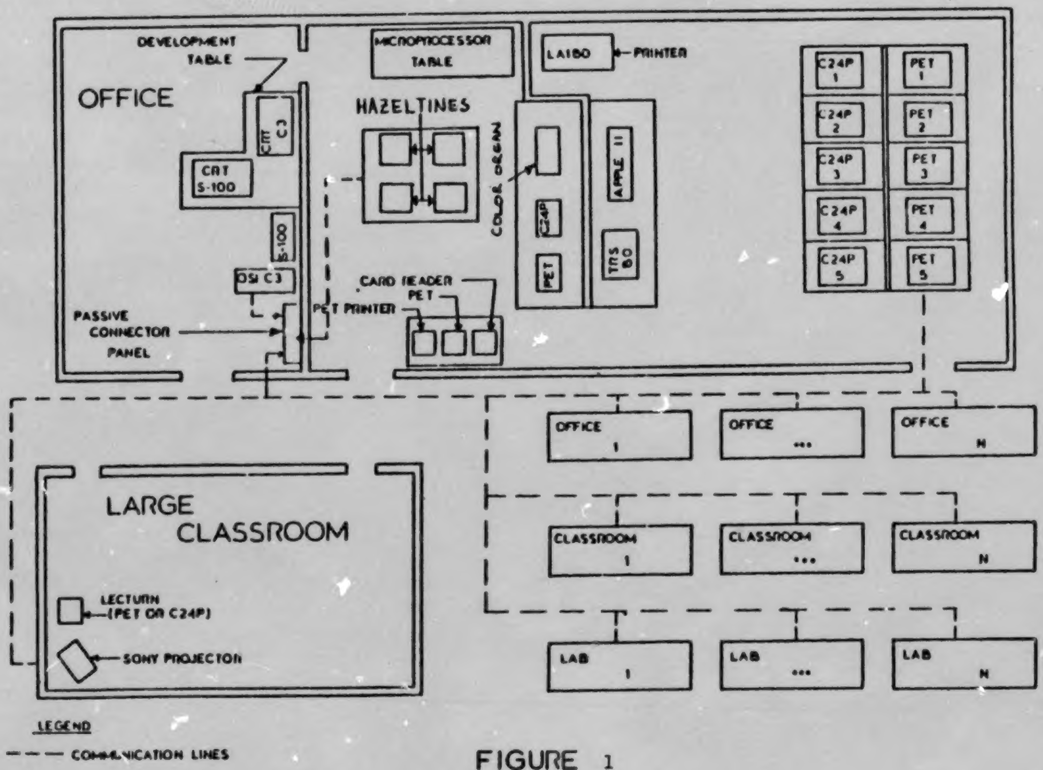

DEVELOPMENT TABLE
OFFICE
CRT C3
CRT S-100
S-100
OSI C3
PASSIVE CONNECTOR PANEL
MICROPROCESSOR TABLE
HAZELTINES
COLOR GREAN
C24P
PET
CARD READER
PET
PET PRINTER
LA180
PRINTER
APPLE II
TRS 80
C24P 1
C24P 2
C24P 3
C24P 4
C24P 5
PET 1
PET 2
PET 3
PET 4
PET 5
OFFICE 1
OFFICE •••
OFFICE N
CLASSROOM 1
CLASSROOM •••
CLASSROOM N
LAB 1
LAB •••
LAB N
LARGE CLASSROOM
LECTURN (PET OR C24P)
SONY PROJECTOR
LEGEND
COMMUNICATION LINES

FIGURE 1

PASSIVE CONNECTOR BOARD

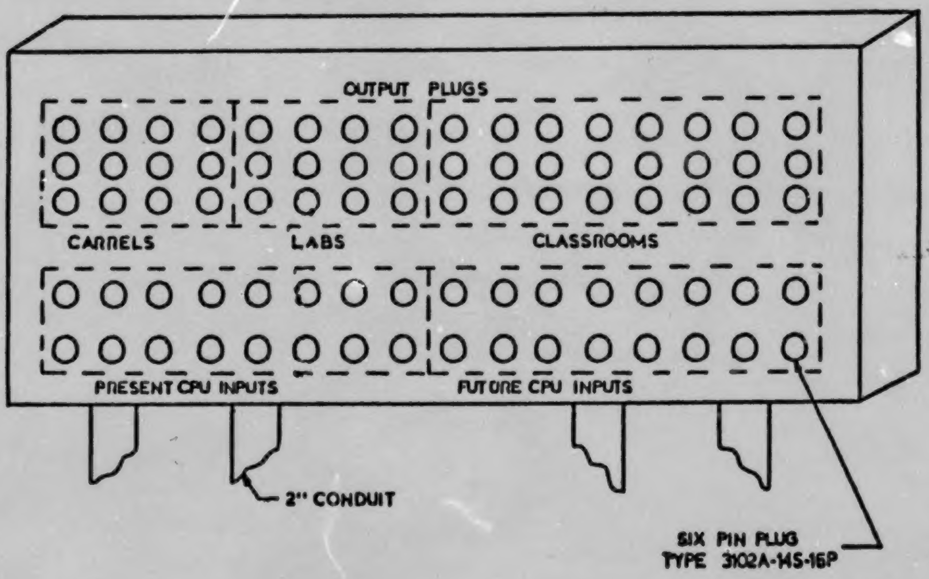

OUTPUT PLUGS
CARRELS
LABS
CLASSROOMS
PRESENT CPU INPUTS
FUTURE CPU INPUTS
2" CONDUIT
SIX PIN PLUG
TYPE 3102A-14S-16P

FIGURE 2

APPENDIX I. SUMMARY OF NETWORK CAPABILITIES

I. Host Software (OSI C3)
  A. Executes in one partition of memory under timeshare
  B. Menu drive for ease of use
  C. Uses user IDs and security levels for command and file protection
  D. Single command deletion of all files under a given ID
  E. Audit trail capabilities to determine station usage
  F. Full range of backup commands via Alloy engineering's twelve megabyte cartridge tape drive
  G. System requires no hard disk repacking
  H. Operates completely independent of OSI's 65u O.S.
II. Satellite Software (PETs, Apples, etc.)
  A. Filing commands include: Create, Delete, Load, and Save
  B. Directory utility also included
  C. File protection via Log-on/Log-off commands, passwords, and read/readwrite file specifications
  D. Host security is extended to network machines
  E. Allows program linking, e.g., one program can load another into memory and execute it
III. Hardware requirements (satellites)
  A. Network software contained in EPROM for 'instant' operation
  B. Requires special interface board
  C. Simple push the button operation to gain access to network software. No POKE, PEEK, CALL, or SY commands are necessary
  D. Any combination of microcomputers can be on the network
  E. Reasonably 32 microcomputers can be put on a network line giving a maximum wait time of 32 seconds or typically 15 seconds or less. If more microcomputers are desired, another partition of memory can be added to the Challenger III system
  F. Micro-net support:
    1. Commodore PET, Apple, Ohio Scientific 4P, 8P, 1P
    2. Coming soon: TRS-80, and Atari 800 and 400

APPENDIX II: DESCRIPTION OF OSI CAPABILITIES

I. Software Features:
  A. Operate under Timesharing
    1. O.S. 65u 9 Digit Microsoft Basic
    2. WP-3 Word Processor
    3. Database Management System - useful for 'hands on' experience
      a. inventory
      b. purchasing
      c. bill of materials
      d. general ledger
    4. Computer Aided Instruction Database
      a. setup and giving of tests and quizzes
      b. statistical analysis of student scoring
  B. Available, but not under timesharing
    1. Digital Research's CP/M operating system
    2. Softech's USCH Pascal/Fortran operating system
    3. Digital Research's MP/M operating system is under development
II. Hardware Features
  A. Triple processors any one of which may be run at one time
  B. High density Winchester Hard Disks (36 or 74 megabytes)
  C. Time Share Operation with up to 16 users/partitions
    1. up to 16 users/partitions
    2. Resource sharing printers, etc.
    3. minimum system 'drag down'
  D. Tape drive backup by Alloy Engineering
    1. twelve megabyte cartridge, or
    2. 45 megabyte magnetic tape which allows IBM data interchange
  E. Full range of hardware products
    1. Votrax voice I/O system
    2. High speed analog I/O system
    3. Security system
    4. Eprom programmer and prototyping board

REFERENCES

Dooley, T., "A Model for Designing a Microcomputer in an Educational Environment." MS Thesis, University of Wisconsin - Milwaukee, 1979.

Herstein, E. "Choosing a Classroom Microcomputer." AEDS Proceedings 17th Annual Convention, May 1979, pp. 328-329.

Thomas, D., and McClain, D., "Selecting Microcomputers for the Classroom." In AEDS Journal, Fall 1979, pp. 55-68.

THE DEVELOPMENT OF LIBRARY AUTOMATION IN A MEDIUM SIZE ACADEMIC LIBRARY: THE U.A.B. EXPERIENCE

Jerry W. Stephens
Mervyn H. Sterne Library
University of Alabama in Birmingham
Birmingham, Alabama 35294
205-934-6360

The use of computers in the educational setting is increasing dramatically. Today's student must be able to use the computer in all areas of educational endeavor. This is true in both classroom instruction and external research. The area of external research is often overlooked in discussion of academic programs and computer technology. As stated by Aaron and Elaine Cohen in "The Quiet Revolution on the Campuses",

> Library science...is moving resolutely into the twenty-first century. The reason is simple: Of all campus institutions, libraries are the chief recipients of the benefits of technology. Computerization, miniaturization, telecommunication, and the audiovisuals are sweeping fresh air through musty halls.
>
> Information is being liberated from the card catalogues, indexes, and miles of stacks. (1)

Computers are necessary tools in the management of today's information resources. Libraries are in the midst of an explosion in the development and application of computer technology to library activities and services. These changes have been developed in most cases with "shoe string" budgets, and are the efforts of librarians, computer center staffs, and administrators.

The University of Alabama in Birmingham (U.A.B.) readily accepted the challenge presented by this evolution. In 1975, U.A.B. became a member of a national bibliographic utility, OCLC, Inc., which provided access to an on-line, shared cataloguing data base. This data base was used primarily in processing items being added to the U.A.B. collections, and was the cornerstone of U.A.B.'s computer applications.

As the use of this service increased, the relative cost of cataloguing a title declined, thus generating a cost savings through the use of computer technology. The timetable for products of national utility began to slip. Other functions in development began to be postponed, and it became evident that it was time to look to the local scene for a more controllable development program.

The U.A.B. Sterne Library System was developed to provide for access and control of the collections of the Sterne Library. The project evolved in two phases in which Phase I became the conversion of a batch acquisitions system to a modular component of the on-line system, and Phase II became the installation and implementation of an on-line circulation control system.

The approach taken was that the acquisitions module should provide tracking and monitoring of the acquisitions process. This module allows tracking from the time an item is selected for acquisition, to the time that item is transferred to the on-line circulation/inventory module.

The acquisitions module also provides the ability to monitor the cataloguing and book processing functions. Items ordered, received, and paid for in the acquisitions module are not deleted at the time of cataloguing but remain on-line until verified cataloguing information for those items is received and added to the inventory model. This means that an item is never lost in the queue.

The OCLC system provides an integral link from the acquisitions module to the circulation/inventory control module. This internal verification provides the library staff with a suitable method for monitoring the flow of items through the various stages of processing (acquisitions and cataloguing). The integral design also ensures that the machine readable record initially created in the acquisitions module is replaced by a full MARC record in the cataloguing process. This MARC record is the basis for the record passed to the circulation module. Thus, the total system is designed ultimately to provide an abbreviated author/title, on-line catalogue for library staff and users.

The design of the acquisitions module was directed toward reducing the manual work load required to order and track materials. This includes the following functions:

1) on-line searching of materials currently requested or on order to prevent duplication of orders

2) on-line item entry and editing capabilities for selected fields

3) automatic grouping of requested items by vendor and purchase order generation

4) on-line processing of invoice and receiving information

5) automatic generation of payments for items invoiced and received

6) automatic generation of claim letters by item for items not received within a predetermined length of time

7) maintenance of fund accounting information and automatic reconciliation of those funds

8) generation of ordering statistics by academic department

9) exception reports from budget and order data bases to provide monitoring of the system

10) year-end fiscal reports for close-out and encumbrances carried forward

The data base access points included in the acquisitions sub-system are indexed by:

1) Main Entry. The Main Entry is a four/four combination of author last name and first word of the title. Leading articles are excluded.

2) Title. The title index is a four/four combination of four asterisks in lieu of the author's last name and the first word of the title. Leading articles are excluded.

3) Item Number. The item number index is an eight digit control number assigned by the system when the item record is created in the new data base. The construction of the item number is the year prefix and a sequential number.

4) Purchase Order Number. The purchase order number index is a ten digit number assigned by the system when the requested items are grouped together on to a purchase order. The construction of the purchase order number is the Julian year/day followed by an alphabetic character as an indicator of the type of purchase order, i.e., A = regular, R = rush, and S = serial renewal. The four digits following the alphabetic character are sequentially assigned page numbers of the purchase order program. The final indicator of the uniqueness of an item is its line item number.

The acquisitions sub-system generates the following forms and reports:

FORMS:

1) Purchase orders which group multiple items to one vendor regardless of requesting department or funding source.

2) Change orders which group multiple items from one purchase order together to show applicable changes.

3) Confirmation orders which are used to confirm the placing of an order.

4) Prepayment orders which create the invoice and payment segments of the record, thus issuing a check to be enclosed with the order.

5) Reference cards which are provided to show that an item has been placed as an order with a vendor. These cards keep collection development personnel aware of the status of items requested for purchase.

6) Claim letters in a series which prompt vendors as to the status of orders and items requested for purchase.

MANAGEMENT REPORTS:

1) Departmental Order Summary which summarizes all outstanding orders by academic department regardless of the funding source or type of material being purchased, and aggregates fund accounting information.

2) Budget Reports which summarize all expenditures and encumbrances in two formats:

   a. by academic department/school regardless of funding source
   b. by funding source by academic department

3) Exception Reports which:

   a. list items in new data base that have not placed with a vendor and the reason placement has not occurred
   b. list items placed in new item data base without vendor assigned (serial renewals not included)
   c. list all items purged to the history tape when completed
   d. list entries to the debit voucher system which have created remittance to appropriate vendors.

The acquisitions sub-system interfaces with other computer applications at U.A.B. The system uses the U.A.B. Purchasing Department's vendor file, which is maintained by the Purchasing Department and is available campus-wide. Security of the file is also maintained by the Purchasing Department.

In a manner similar to its use of the vendor file, the library system communicates payment information to the Accounts Payable system of U.A.B. The data entered into the library system are used to create ledger entries in the general accounting system, and to produce checks payable to vendors. This system's totally programmatic access ensures security and control points when interfacing to these centralized functions.

The circulation sub-system was

designed to monitor and control the location and use of the collection of the Sterne Library. The system involves a continuous and constant monitoring of the collection via the available transactions of course reserves, check-in, check-out, and holds.

The system allows the staff to assign a unique control number to the volume being added to the inventory. This number produced as an OCR label is attached to each piece as it is added to the inventory. The use of OCR was chosen because of the technology and because of the readability of the label with the eye. Each patron is also assigned a unique control number, also in OCR format. This allows the recognition equipment to be used for transactions and maintenance. Use of bar codes labels is possible with only minor changes in hardware.

The purpose of the patron data base is to allow for control of the user population in an efficient and effective manner. The system monitors all users as to their validity and status. The patron data base allows for multiple patron type and institution codes, and for a variety of options based on this flexibility. The patron data base allows for monitoring and controlling patron usage via nightly batch programs. Via these batch programs, the system performs the functions of building complete patron records, sending overdue and excessive fines notices, and validating patrons as applicable. The system allows the operator access to one centralized data base on all members of the user population.

The purpose of the patron search function is to have a second index to the patron data base. The patron search routine allows the system to provide the operator with information regarding a patron by name.

The course reserve function is a mechanism for the modification of the retention period of permanent and transitory items in the collection of the Sterne Library. This monitoring and control mechanism is used to maintain the reserve book collection. Transitory items placed on reserve are entered under a unique institution code which makes them distinct from the permanent collection. The reserve function simply modifies the routines of the check-out transaction. This modification allows the operator to limit or modify the retention period of an item.

The system is designed to handle the continuous check-out of up to fifteen items per screen. It verifies whether a patron is allowed to check out material, as well as whether the status of a particular book allows it to be checked out.

By entering the patron's ID number, the operator is able to verify the status of the patron. If the patron is valid, the system returns the patron's name, address, and the limit of books the patron is allowed to check out. If the patron is invalid, the system automatically returns the patron screen, which displays the reason the patron is not valid and consequently not allowed to check out books.

If the patron is valid, or if authorization is granted, books are checked out by entering the book ID number into the appropriate field on the screen. If this is done with the OCR wand, the book ID numbers appear in the column down the left side of the display terminal. If this is done via the keyboard, the operator must be careful to enter each number in the proper field.

The purpose of the check-in function is to allow items to be returned to the collection. The items being returned to the collection via the check-in screen have destination assignments. The check-in display allows the operator to monitor and route items to their proper destinations. The check-in function also allows for monitoring of fines, lost books and holds by patron. The check-in function updates the book data base and the patron data base via a single transaction.

The purpose of the book record data base is to group in one location a complete shelf list of all the Sterne Library holdings. This is done and arranged by different indexes. The data base is available on-line to assist the operator in determining the status, type, and restrictions of any item in the data base.

This data base is composed of two basic types of materials. One type includes all of the permanent holdings of the Library. The second type includes transitory items entered into the holdings on a temporary basis for a particular use or purpose. The data base is a complete compilation of all Library holdings.

The purpose of the book search function is to allow the operator to query the holdings data base by author/title/call number/OCLC number. The indexing of the book record data base is such that the system searches under the following options:

1) Main Entry. The first twenty characters of the corporate main entry or the last name (twenty characters) and first initial (one character) of the personal author.

2) Title. The key word strategy which is designed to accept up to six key words of the title in any order. The indexing is by six fields of up to twenty characters each. Each key word must be in a separate field.

3) Call Number. A field-by-field call number search. The system requires a minimum of two fields of the call number to complete a search. The call number search omits special prefix tags, such as REF, Juvenile, Phono Records, etc. The call number indexing is arranged in ten fields of eight characters each.

4) OCLC Number. A search by the

exact and known index to the record in question. The OCLC number is a nationally accepted record identification report.

A hold is a request for an item that is currently circulating. The hold function allows the operator to request that an item be held for any number of individual patrons without having the actual item in hand.

The circulation sub-system produces the following forms and reports:

FORMS:

1) Overdue Notices: On a predetermined schedule, lists all problems with an individual user.

2) Activity Purge: Itemizes all transactions by the user that are considered completed transactions.

MANAGEMENT REPORTS:

1) Report of daily activity processed by the system

2) Report of invalid users and reason for that status

3) Report of usage by type of user, by institution, by type of material, and by LC classification number

4) Report of material declared lost or withdrawn

5) Report of activity purged, users purged, and holds remaining

6) Exception report of item records found on OCLC which have not been added to the data base and the reason

The circulation sub-system has allowed the Library staff to become more aware of the diverse needs of the users. The system has freed the staff from mundane tasks such as filing and pulling cards, hand-addressing overdue notices, computation of circulation statistics, etc. The time previously spent with these tasks is now devoted to meeting the needs of the user population.

The automation project, though not complete, has provided the user population and the Library staff with numerous benefits, including more accurate information about the collections, the activities of collection development and control, and a greater awareness of what is contained in the collections. While either subsystem presently can stand alone, developments for the future include a combination of the data bases to provide for integrated access to the information, as well as full access to the additional secondary indexes. These developmental changes are proceeding as rapidly as possible given the current constraints.

The Sterne Library at U.A.B. has used the technology, as suggested by the Cohens, for "shrinking the backroom functions and adding to the services." (2) The librarian must respond to these technological changes to provide the best services for the benefit of the academic community.

---

(1) Aaron Cohen and Elaine Cohen, "The Quiet Revolution on the Campuses," The Chronicle of Higher Education, November 25, 1981, p.56.

(2) Ibid.

Special Session

MICROPROCESSORS IN THE WHOLE OF ENGINEERING EDUCATION THE CONTEMPORARY MICRO LABORATORY

Sponsored by: IEEE-CS
Chaired by: Thomas Brubaker
Colorado State University

ABSTRACT

Panel members will discuss the utilization of microprocessors in all aspects of engineering education. Classroom techniques and implementation concerns will be addressed.

Participants

Tom Cain
University of Pittsburg

John Gowdy
Clemson University

Ronald Krutz
Carnegie-Melon University

Terry Smay
Iowa State University

Special Session

COMPUTERS FOR PUBLIC PUBLIC CONSUMPTION

Chaired by: Ann W. Lewin
Capital Childrens Museum

ABSTRACT

The computer revolution was announced 30 years ago, but only dropping costs and the creation of powerful microcomputers have made it a populist revolution. Computers are now everywhere, and many persons have suggested that they will enter society through homes, offices, arcade games, public settings, and everywhere other than the schools. In this session, authorities will discuss whether the microcomputer does in fact hearld the beginning of the end of school, and they will describe powerful non-school applications.

What Should We Be Telling The Public About Computers?
Paul Pangaro will discuss the myth of AI and new directions for educating the public about what computers can and cannot do.

Why Education Cannot Take Place in Today's Schools.
Ramon Zamora will discuss the factors that shift traditional education away from the schoolroom and into the house. He will preview a new software series created by an artist and developed using a videotape (to be shown during the session) as the conceptual and programming specification.

When You Must Have a Teacher.
Dr. Arthur Luehermann will talk about situations in which a teacher is essential to faciliate learning with computers and those situations in which a computer cannot do the job.

Where does the economically disadvantaged kid get his? If the school is on its way out, what happens to the generation of kids that are there now?

Participants

Dr Arthur Luehermann, Physicist
Computer Literacy Incorporated

Paul Pangaro
Capitol Childrens Museum

Ramon Zamora
ComputerTown, USA

Project Presentation

SOFTWARE DEVELOPMENT

Chaired by: Alan Roecks
Education Service Center

Frank C. Brandt
James W. Garson
Carin E. Horn
Barbara H. Knapp
Frank Rinehart
Lynn Rosenthal
Helen J. Schwartz

ABSTRACT

Development Of The Video-Computer Courseware Implementation System At The University of Utah

Richard C. Brandt and Barbara H. Knapp,
VCIS Software Group, University of Utah,
Salt Lake City, Utah 84112

The development of the Video-computer Courseware Implementation System started in the summer of 1978. Initially lessons were implemented as Pascal programs but we soon realized we needed to create a system that would permit the implementation of most of the courseware without programming. Writing the courseware in Pascal isolated the authors from their lessons and increased the implementation and modification time.

Four design objectives were adopted: the system should permit varied instructional strategies, reduce authoring and implementation time, permit easy lesson modification, and provide information needed for lesson refinement. To achieve these objectives four major design decisions were made. First, the command structure was to be heirarachial; queries or menus were to always direct the user. Second, the command language was to be easily understood. Since the majority of users of the implementation system were not to be programmers, programming terms were not used unless they agreed with the ordinary meaning of the word. Third, the system was frame oriented so the editors displayed what the student would see. Fourth, the lessons were to be based on data files rather than programs and as many of these datafiles as possible were to be editable.

In the current implementation system the author or a team of authors determines the conten of the lesson and outlines it on paper. Editors for text, graphics and animations are available for creating the contents of the frames which the student will see. A program is available for selecting video segments from tape or disc. The structure of the lesson is defined in a separate program. Lessons can be built and tested in segments, modified, and then combined. When the lesson is presented to the student, the student's answers are compared to the author's anticipated answers to determine which material the student will see. A student's path through the lesson can be recorded and used to evaluate the lesson. Instructors can specify whether students can back up in a lesson, if data is to be collected, and if video material is to be used.

ABSTRACT

ZGRASS: A Simple Language for Graphics Courseware

James W. Garson, University of Houston,
Central Campus, Houston, Texas 77004

The development of graphics software for education is a time consuming process, especially if the programmer wants to make use of animation. Languages which are widely available for microcomputers have a number of serious limitations which make graphics programming difficult.

Over the last ten years, Tom DeFanti and his co-workers at the Electronics Visualization Laboratory have been developing ZGRASS, a system which makes graphics programming relitively easy. The ZGRASS language allows amateur programmers to create and control complex animated images with modest effort.

In this demonstration, we will introduce the features of ZGRASS which simplify the process of software development for education. Along the way, we will explain how similar ideas can be adapted to more widely known microcomputers.

We will begin by reviewing some of the graphics commands of ZGRASS, and the flexible methods it uses for controlling color.

Next we will discuss the simple and natural methods ZGRASS uses for the modular development of images and programs. Finally, we will demonstrate ZGRASS's most unusual feature: its ability to run several programs in parallel. The importance of this feature in developing modular animation will be explained, and a number of programs which use it will be presented, including one which teaches graphics algorithms.

The demonstration will close with short excerpts from educational video tapes developed on the ZGRASS system.

ABSTRACT

Voice Controlled CAI

Carin E. Horn, Doctoral Candidate. North Texas State University, Denton, Texas, and William D. Wagers, Systems Analyst, Scott Instruments Corp., 1111 Willow Springs Drive, Denton, Texas

Microcomputer-based voice entry terminals are eliminating the need for the CAI student to translate mentally derived solutions into keyboard manipulations while facilitating student/machine communications. The aim of this presentation will be to demonstrate a voice-controlled instructional strategy dependent upon speech recognition and voice entry techniques using an Apple II Plus System.

The demonstration software, called The Prompt Machine, employs prompting procedures appropriate to the acquisition of fundamental proficiencies in any discipline. The lessons to be reviewed allow for teacher-authored question sequences and voice-entered student responses in both a tutorial mode and a review mode. Learning tasks incorporate the following procedures:

1) stimulus term or phrase (question), which may include supplemental information, is visually displayed to the student for a fixed interval;
2) response term or phrase (answer) is visually displayed and a spoken response is solicited;
3) overt practice (vocal entry) of the response term (answer) is input by the student.

This sequence constitutes the tutorial mode during which the learning task is presented. The review mode follows and incorporates a summative evaluation of the student's performance. The test mode is a teacher option which is available for evaluating retention.

Audience participation and feedback will be encouraged, evaluative results of variuos field responses to the system will be discussed, and a written description of the project will be available to participants.

ABSTRACT

A CAI Authoring Environment for Non-Programmers

Frank Rinehart, PhD, Lynn Rosenthal, PhD, College of the Virgin Islands, Chorlote Amalie, St. Thomas, US Virgin Islands 00801

Development of attractive screen layouts in CAI programs is time consuming, particularly when upper/lower case is used, pictures included and text placed in a variety of positions, possibly surrounding pictures. Systems such as Apple Pilot save programmer time by accepting text from a non-programming Author, but the results are rigidly limited.

A SCREEN-BUILDER/RUNNER system has been developed in which an Author designs a tree sequence of events and text/picture displays specifies variable I/O required and any processing necessary. SCREEN-BUILDER provides the Author with a blank screen and freely moving cursor. Control Keys allow case and cursor control, insertion/deletion functions as in a word processor. The Author assigns the screen a number, indicates preceeding and (alternative) following screens, and specifies optional clearing of (portions of) preceeding screens to be done before display. Picture-are created by the Author using the Apple Animatrix system. A programmer writes short subroutines which control printing of pictures, I/O and processing. SCREEN-BUILDER allows picture subroutines to be invoked or previous screens to be displayed while designing a new screen thus eliminating positioning problems. This is vital in a display sequence which gradually builds up a complex display as interaction occurs. A shell RUNNER program loads the pre-defined screens and displays them. With each display the corresponding subroutine is called and the following screen defined.

The system seems to provide a "friendly, supportive" environment program development without rigidity or restriction, and it is hoped that much lead programmer time will be replaced by faculty/student-helper teams. At writing, the system is complete and has been used for program development, but some bugs remain in system functions for adding/modifying displays.

ABSTRACT

A CAI Program for Literary Analysis

Helen J. Schwartz, English Department, Oakland University, Rochester, Michigan 48063

MARSY/EBB is a series of chained computer programs developed by Helen Schwartz, at Oakland University, Rochester, Michigan, to help students with an analytical problem within a discipline.

The educational goal includes the following specific objectives: to help students 1) generate new ideas and 2) refine those ideas for communication to others, in such a way that the students will 3) generalize their understanding of disciplinary procedures and methods and internalize them by 'tracing' the development of their ideas. The three parts of MARSYEBB implement these objectives with: 1) a tutorial tailored to a particular discipline, which helps the user to generate new ideas. (At present, the tutorial is designed to help a user develop an analysis of a character in a literary work.) 2) The Electric Bulletin Board (E.B.B.), which helps a student refine his/her ideas by communicating with others in a protected environment. When the user is satisfied with his/her ideas, they are posted as a "notice" on the E.B.B. and signed with the user's pen-name. After creating a notice, the user gets access to the E.B.B. where s/he can see any notice, add a comment to a notice, and/or see any notice. 3) The student's individual folder, which files old notices and comments, so that s/he can 'trace' the development of his/her ideas and thereby generalize and internalize the process of analysis. When the user revises or replaces a notice, the old notice (and any comments on it) are removed from teh E.B.B. and filed in the user's individual "folder". The user can then have the folder printed out.

For research and administrative purposes, MARSYEBB records the user's activities at each session(whether creating or modification of a notice, seeing notices and comments, or adding comments). This section can be removed if not needed.

The current application is designed for an introductory class in literature and provides a tutorial for developing a character analysis. Because the component programs are modularized and self-contained, they can be modified for different educational designs: within literary analysis (for example, a tutorial on plot analysis), or in other disciplines.

Equipment necessary for MARSYEBB includes 1) a microcomputer (with 48K in main memory and 1 or 2 disc drives) run under CP/M (for standardization) and with Microsoft Basic; 2) a printer. (The programmer has used an APPLE II Plus with Microsoft Softcard. Either a 13-sector or 16-sector disc can be used--with more memory available on the 16-sector disc with the 3.3 DOS.)

This series of programs is available from the author: Helen J. Schwartz, English Department, Oakland University, Rochester, Michigan 48063.

Project Presentation

THE COMPUTER AS A TOOL IN UNDERGRADUATE COLLEGE CURRICULUM AND BEYOND

Chaired by: Lawrence Jehn
University of Dayton

Ralph M. Deal
Margaret Ellis
Alice Graghard
Dale M. Johnson
Elaine Reber
Dale Schoenenfeld
Sara Stone
Susan Ward
James Watson
Nancy Watson

ABSTRACT

Modelling Chemical Reactions in PASCAL

Ralph M. Deal, Kalamazoo College, Kalamazoo, Michigan 49007

Chemical kinetics mechanisms have been modelled on computers in many languages for educational as well as research purposes. I have used FORTRAN, CSMP, NDTRAN, and DYNAMO to do so and am now working on a Pascal implementation that will seem "natural" to chemists so that they can model mechanisms of interest to them by simple modification of the Pascal model provided.

The current version is passable and produces reasonable graphical output. Eventually, the program should prompt the computer-naive chemist to give the pertinent information in the chemist's language. The more general the mechanism, the more complex the program unless it is written carefully. I hope to have a prompted system that will work well in which the Pascal coding can be explained simply to a chemist not "literate" in Pascal.

ABSTRACT

Computers in Instruction at Seattle Central Community College

Margaret "Kit" Ellis, Seattle Central Community College, 1701 Broadway, Seattle, Wash. 98122

A very successful college-wide program for the use of computers in instruction has been established by faculty at SCCC. With the whole-hearted support of the administration, the use of microcomputers has been introduced into a large and still growing number of classes.

The program had three main objectives:

1. to initiate a faculty training program in the use of computers in education;
2. to obtain appropriate computer resources (hardware and software);
3. to develop computerized record keeping systems for the basic skills labs.

The Computer-Assisted Instruction and Computer-Managed Instruction (CAI/CMI) Project has been in operation for two and one-half years and has produced several major accomplishments:

1. More than 80 faculty members have completed at least a ten hour in-service course on the use of microcomputers in instruction.
2. CAI is currently being used by more than 40 instructors in at least 80 course sections annually.
3. CMI is used in class record keeping for more than 1200 students in basic skills labs. The early warning and intervention features result in an increase of ten percent in the completion rate of mathematics laboratory students.
4. Five new microcomputer-based courses were established in the curriculum to serve the needs of students in science and mathematics.
5. A microcomputer facility is available to students and faculty nine hours a day, five days a week.
6. A library of over 300 instructional programs has been established.

The success of the instructional computing program was the result of an ideal combination: a full-time faculty coordinator; enthusiastic faculty cooperation; supportive administration; and a well con-

ceived plan with clear goals, objectives, timelines, and evaluation criteria.

ABSTRACT

Learning How to Bank at Home

Alice Gragnard, Sara Stone, College of Communications, University of Tennessee, Knoxville, Tenn 37916

In January 1981, the United States Bank of Knoxville became the first bank in the United States to initiate a home banking service. Thus far, the inovative concept has drawn some 400 bank customers. The in-home computer banking service can be used not only for personal finance, but also for many other services, educational or entertainment needs. Participation in the Bank at home service requires a computer, a color television monitor and a telephone. With these, the customer has access to his checking and savings accounts, and can pay some bills, send and receive mail electronically, obtain the latest updates on certificates of deposit, savings and money market rates, and have access to a wealth of news, weather, sports and financial information on the news and financial information network.

Our project presentation will center around an actual demonstration of the United American Bank System, including how to use home banking and Compuserve services. An instructional slide presentation on the use of the computerized home banking system will be included.

ABSTRACT

TEST - Test Evaluation & Score Tabulation

Dale A. Schoenefeld, PhD, Division of Mathematical Sciences; Dale M. Johnson, PhD, Division of Professional Studies, University of Tulsa, 600 S. College, Tulsa, OK 74104

This presentation describes the rationale for the utility of TEST program. Program TEST developed and recently implemented at the University of Tulsa specifically for addressing the following issues:

1. Many faculty (public school and small university) do not have access to test scoring and analysis from institutional data processing centers and are therefore "on their own" relative to automated test scoring and analysis.
2. Much of the available software is machine dependent and hardware bound relative to exam response input.
3. Due to the almost universal faculty access to various hardware configurations, there is a need for generalized standard scoring and analysis program that can be executed on microcomputers as well as large mainframes.
4. In light of increasing computer literacy among educators, a responsible expectation of faculty is to be able to manipulate data files without being responsible for sophisticated programming modification
5. Testing software should provide optional output that supplements simple scare reporting for the instructor who is "serious" about evaluating the assessment instrument itself.

With these parameters providing direction, TEST was planned, coded in American National Standard Subset FORTRAN (1977), tested, and documented. The presentation will address: (1) how TEST satisfies the above issues from a programming/technical and testing/psychometric perspective, (2) how to implement the program (code will be provided), and (3) how to interpret the output (example output will be provided for the participants).

ABSTRACT

University of Missouri-Columbia College of Agriculture Microcomputer Facility

Susan Ward, Elaine Reber, UMC Department of Agricultural Economics, 200 Mumford Hall, Columbia, MO 65211

The College of Agriculture received funding last summer for a microcomputer laboratory to provide students and faculty with the resources to:

1. Teach computer programming in a hands-on environment
2. Encourage the development of computer assisted instruction (CAI) programs for current and future courses
3. Provide a facility where students, faculty, and staff can develop programs to satisfy research, teaching and extension needs.

This laboratory has recently been completed and is equipped as follows:

1. 14 Apple II plus microcomputers and monitors
2. A Paper Tiger graphics printer
3. A Hiplot plotter
4. Two digitizers
5. One Corvus hard disk (20.5 Mbytes)
6. Four 5 1/4 inch floppy disk drives
7. One D.C. micromodem

The available system software includes Visicalc, CCA (data base manager), SUPERTEXT (text editor), APPLE PLOT (useful for developing courseware), ASCII EXPRESS II (uploads and downloads datasets from Apple to Amdahl), numerous statistical programs, Young Farmers Record Keeping System, etc. FORTRAN, BASIC, and PASCAL languages are also available on the system.

A series of College seminars is planned to introduce faculty and staff to this new and exciting facility.

Presently, 250 students in the College

of Agriculture are enrolled in an introductory course in computing. These students use the microcomputer facility to learn basic programming skills and become familiar with many of the programs commercially available on microcomputers. They are experiencing computing in a hands-on environment and acquriing marketable computing skills that will benefit employers and our basic food-fiber industry. The College expects ultimately to teach at least 300 students a semester with this course.

ABSTRACT

Using a Microcomputer Network for Instruction of All Ages: Six to Sixty

Dr. James Watson, Jr., Assoc. Professor, Department of Physics & Astronomy; Mrs. Nancy Watson, Assistant Professor, Department of Burris Laboratory School, Ball State University, Muncie, IN 47306

In the fall of 1980, the Department of Physics and Astronomy purchased a network of twelve Radio Shack Model I, Level II Microcomputers. These twelve machines are connected by a network controller to a two disk, 32 K machine used by the instructor. Through the controller, the instructor can give each of the other machines the same or different programs.

The network has been used with many different classes, groups and ages. The network is used in every lab period for the general physics course for elementary education majors. It is the basis for a computer literacy lab in the general physics program. It is used as a simulator and backup for the department's solar energy course. Other courses use it for general calculations and analysis. It is also used during high school class visitations.

Due to the unique situation of having Burris Laboratory School, a K-12 public school, on campus, students of all ages are using the network. The students are taught how to use microcomputers and simple basics of BASIC. Software and general computer use are introduced to the students on the network.

Burris has about two Radio Shack microcomputers, an adult education class is taught using the network. The students have varied reasons for taking the class: retired people who are just interested in computers, professionals who are interested in using the computers in their businesses and a few who are just curious.

In the past two years, the network has been used to instruct people of all ages and with different needs.

Tutorial

EDUCATIONAL SOFTWARE: ITS CREATION AND EVALUATION

Jerry Johnson
St. Olaf College

ABSTRACT

This tutorial will feature a discussion of the NCTM/s recently published comprehensive guidelines for the evaluation of computerized instructional materials. These guidelines provide a practical aid for both users and creators of instructional computer software and assume no programming experience. One of the authors of the guidelines, Jerry Johnson, will lead the audience of this tutorial through the five principle tasks of 1) obtaining software and its documentation, 2) evaluating the materials obtained, 3) making a decision regarding the acquisition of the software, 4) establishing a system for helping all potential users to locate programs of instructional interest, and 5) modifying or improving the programs.

Evaluation of actual programs will also be included in this tutorial. Every educator concerned with instructional delivery systems, classroom teachers as well as administrators, should find this tutorial useful.

Computer Science

A MORE ANALYTICAL APPROACH TO TEACHING THE INTRODUCTORY PROGRAMMING COURSE

Lionel E. Deimel, Jr.
David V. Moffat

Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27650
(919) 737-2858

INTRODUCTION

Careful, systematic problem analysis is now a major feature of introductory programming courses. The authors believe, however, that a more analytical approach to programs themselves is necessary. We will explain why we believe this and will propose a structure for a course using this approach. We will examine how such a course might improve on more traditional ones. Finally, we will consider special requirements for the course and some research implications of our proposal.

INTRODUCTORY PROGRAMMING COURSES

The typical introductory programming course for computer science majors attempts to convey the notion of programming as problem solving. The distinction between algorithm and program is emphasized, and the need for design, usually in a top-down fashion, is stressed. A particular high-level programming language is taught, and the student ultimately learns to synthesize programs and procedures from the statements of this language. We will examine the origins of this course, the problems its teaching presents, and alternative approaches which might be available.

The conventional wisdom once was that the teacher of an introductory programming class needed to get his students programming immediately, in order to engage their interest. Textbook authors and their publishers emphasized how their books facilitated this activity. Prefaces and promotional literature often contained statements such as "we have found that it is important to begin work with the computer as soon as possible--preferably immediately after the first lecture"[5] and "the student is able to write a *complete Fortran program after reading the first chapter*" [15]. In fact, many programming texts seemed to promote the idea that programming was fundamentally easy--one only had to learn a programming language to engage in it. Programming courses were, in fact, merely programming language courses.

The recognition of the "software crisis" and the widespread acceptance of structured programming altered the conventional wisdom. Programming as a *problem-solving* activity which included a substantial design phase came to be emphasized. Every new textbook title had to include the phrase "problem solving," "structured programming," or (preferably) both. The more modern approach is well-expressed by Dyck, *et al.* [8]:

> ... the authors feel that this approach [getting the students on the computer immediately] encourages poor programming practices. ... We do agree that students should get some early exposure to the computer for motivation, excitement, and a sense of achievement. However, we would argue that some minimum level (say a few lectures) of instruction on algorithm design should be undertaken first.

The experience of the authors suggests that this point of view is often difficult to implement. Programming language courses *have* generally evolved into problem-solving courses, but the temptation to loose the students on the machine early and the demands of students to begin programming quickly are difficult to resist. Despite the instructor's resolve to emphasize concepts, he is constantly frustrated by the students' tendency to focus on the concrete--the programming language--rather than on the abstract notions of problem solving and algorithm development. This has lead some to advocate separating conceptual and practical aspects of programming into distinct lecture and laboratory courses [9]. This idea seems not to have caught on, no doubt because of the many obvious implementation problems. (Prather and Schlesinger [13] and Dershen [7] describe some of these problems in relation to courses they clearly believe employ this scheme successfully.) Meanwhile, textbooks continue to make boasts such as that they are "organized so that the beginner can start running programs almost immediately" [4].

The emphasis on problem solving is the most conspicuous, but by no means the only change which has occurred to the beginning course. Despite the continuing presence of syntactical minutiae, an increasing number of other concerns vie for the student's attention. The goals of the course have expanded in response to the increased responsibilities required of the professional programmer.

Today programs are expected not merely to "get the right answer," but increasingly are not even written to "solve a problem" in any simple-minded sense. Instead, they are expected to be "correct," "robust," "user-friendly," "well-documented," and so forth. This places a heavy, often freely-acknowledged burden upon the student [6]. The growing list of virtues to be possessed by his programs is especially troublesome for the student, as so many desiderata involve matters of style and aesthetics or grow out of the pragmatics of a workplace environment of which he has little knowledge. The authors have found that the average student often becomes confused and frustrated under these conditions. In large measure, this seems to be because he does not understand what is being required of him and because he is being asked to master so many skills at once. In light of the need for computer professionals, it is unfortunate if potentially competent programmers are being discouraged by introductory course demands.

There are variations among introductory courses, though judging by the selection of available textbooks, the variability in approach is not substantial. Recent texts reflect the understanding that programming is fundamentally problem solving and that the responsibility of the programmer extends beyond producing code that "works." These books introduce development tools other than conventional flowcharts--pseudocode, structured flowcharts, data tables, etc. Reflecting the need for more efficient and effective programming, they contain explicit discussions of style, testing and debugging, and the like. Nonetheless, the most significant differences among programming textbooks, aside from the language used, continue merely to be the selection of language features treated, the order of presentation, and typographic details such as the use of shading, color, and fonts.

Teachers of programming (we need not limit ourselves to those in university computer science departments) must ask themselves what strategies are likely to meet professional and academic goals without placing unreasonable burdens on the student and likely discouraging many from pursuing computer careers. One possibility is to replace the introductory programming course with an introductory *computer science* course. Several texts are available for doing this [3, 10, 17]. The strategy is unlikely to be useful outside a department of computer science, however, and its effectiveness, either as a programming introduction or with respect to more generalized goals, is not yet established, despite its intellectual appeal. Significantly, both Curriculum 68 [1] and Curriculum '78 [2] include introductions to programming rather than to computer science.

An obvious way to relieve the burden on the beginning programming student would be to take a spiral approach to the teaching of programming. Over several courses, the proper goals for programs might be successively refined. As the student gains experience, he would be required to generate successively better designed and better written programs. It is this technique, in fact, by which we learn to write in school--grade by grade our writing skills are refined and our objectives become more sophisticated. At least two difficulties arise with an attempt to use the spiral technique. The first is that there may not be a sufficient number of programming courses available in the curriculum to implement the necessary carefully planned and controlled program of instruction. Secondly, the student may develop bad habits which are difficult to break. As Schneider has pointed out, an advantage of presenting topics such as algorithmic development and style from the beginning is that good habits are instilled simply because the student has seen programming done no other way [14].

Introducing some other language *before* presentation of the major language of the course is another option which has been used from time to time. (The authors' department formerly did this.) This language may be an assembly language for a real or hypothetical machine, and the purpose of its introduction is to foster increased understanding of what follows. The intriguing book *Karel the Robot* by Richard Pattis [12], provides a thought-provoking variation on this strategy. The book introduces an imperative, Pascal-like language which directs a robot through his checkerboard environment. This approach represents a rather indirect way of giving the student the proper "feel" for programming, one for which some success is claimed and about which some controlled research is certainly indicated. Nonetheless, the very indirection of *Karel* may lead to some confusion. Further, the effectiveness of such a technique in the hands of the less skilled instructor may be greatly reduced.

Actually, the authors believe that the introductory programming course may need a radical restructuring rivaling that which grew out of the software crisis. We now propose a substantial break with the traditions of this course in a way we think many computer scientists will find quite plausible.

TEACHING A DISCIPLINED, ANALYTIC APPROACH TO PROGRAMS

The stress on algorithms and top-down design has undoubtedly produced courses which have tended to de-mystify programming for the student. Instead of concentrating on the synthesis of programs from primitive syntactic components, the emphasis has shifted to analyzing problems into successively more basic atoms. This shifted emphasis leaves two broad areas of difficulty untreated. First, the problem to be solved is usually taken as given, and the nature of the proper interplay between program and user is insufficiently explored. (Textbook problems generally present a very simple view of the world.) Second, the implication is usually present that the production of a program from an algorithm is itself a straightforward, algorithmic process. This too is a simplification. Both these areas of difficulty can be treated, we assert, by presenting *programs* as well as *algorithms* in an analytical fashion. A disciplined, multi-phase exploration of both notions may provide a significantly better understanding upon which to base computer science education. In this section, we describe how such an approach might be applied to the introductory programming course. Later, we will explore the rationale for this approach and some of the implications of it. Our proposed course comprises four distinct phases:

1. Gaining experience as a user.
2. Analyzing programs and algorithms.
3. Modifying programs.
4. Designing and implementing programs.

In the first phase, which may last for two weeks (we assume a semester schedule), students run "canned" programs, observe their behavior, and evaluate their strengths and weaknesses. They learn what computers and programs can do, how they are used, and what input and output look like. Some of these programs may be quite complex; all should be conspicuously useful. The students learn to employ user manuals and become familiar with computer equipment.

In phase two, lasting perhaps five weeks, students are introduced to programs in source-language form and to algorithms. The primary activities during this period are reading and hand tracing of programs. The notion of algorithm is introduced and its relation to concrete programs explored. A major goal of this phase is to teach the programming language being used (*e.g.*, Pascal). This is done largely by extracting language details from properly organized and documented programs rather than from a conventional text. Students ultimately review and evaluate programs for their adherence to basic stylistic guidelines. As they learn the language, students also learn to identify common algorithms. They run some of the programs examined, in order to verify the correctness of hand tracing.

During the third phase, students modify and run existing programs. This lasts for about a month. The major purpose here is to begin coding, putting knowledge of the language to the test of self-expression. At the same time, students learn the basics of testing and debugging.

In the fourth and final phase, students at last design and write programs on their own. About a month will be allotted to this. Based on their experiences of using and anlyzing programs, students learn to write program specifications, to design programs in a top-down manner, and to implement their designs. They can apply, by analogy, their knowledge of the language, of common algorithms and strategies, and of organization and documentation to "new" problems in order to produce entire "original" programs. They learn basic development strategies and further explore testing and debugging.

We feel that the term "disciplined" applies to the approach we have described because the technique assumes a thorough exploration of and concentration on the activities performed during each phase. Students are not allowed to jump ahead, even though the temptation to do so is very great. The activities pursued in each phase are of course carried forward into the next phase.

WHY A NEW APPROACH?

The primary products of the student in the usual first programming course are programs. Yet much of the student difficulty experienced seems to arise from the failure to grasp, in any fundamental way, just what a program is, how it should behave, and what it should look like. Since a good program is actually more than simply a straightforward implementation of an algorithm, it is not enough to preface program-writing exercises with instruction in algorithm design. What is needed is an approach which attempts to foster more intuition about the nature of programs before any are written, thereby providing motivation for such requirements as robustness, proper style, and documentation. In particular, the student should develop the habit of viewing programs and procedures in functional terms, relating inputs to outputs. The many needs of the user and the programmer's obligations to him must be appreciated. The student needs to understand that programs are embedded in complex hardware, software, and human systems which have to be taken into account by the programmer. Especially does the student need to recognize that programs, even those under development, should be documented, readable, and maintainable. In the usual course of study, appreciation of the above issues is slow to develop and matures over a long period. The method of presentation outlined in the previous section is aimed at expediting this maturation.

The key, we suspect, to developing better notions about programs is the *reading* of programs, not merely the casual perusal of textbook programs, but the serious study of *nontrivial* working programs. It has been more than a decade since Gerald Weinberg implored programmers to seriously read programs [16], but there is little indication that anyone has paid more than lip service to his plea. Programs are still in large measure what Kernighan and Plauger have called "write-only" [11]. Computer science has no reading tradition. To put it bluntly, most programmers are only half literate--they can write but cannot read, and this situation is the result of their training.

It is surprising that our semiliteracy is not seen as remarkable. Programs are the primary artifacts generated by programmers. They are bodies of text in what is quite properly called a language. They constitute the basic literature of programming. Is it not surprising that we attempt to teach composition in unfamiliar languages without encouraging reading knowledge of the literature in those languages? When we take seriously the notion of a programming language as a *language*, the "naturalness" of our proposal becomes apparent. One of the fastest methods of learning a second natural language is, after all, immersion in a culture utilizing that language.

Looking at particulars, one can begin to see how the analytic approach to programs can help develop important attitudes and skills. The first phase (running canned programs) gives the student a clear and *practical* idea of what programs are for and what a "user" is. It is important that the student feel comfortable with computers as soon as possible and that he become sympathetic to those who employ the products of programmers.

In the reading phase, the student extracts details of the language and of algorithms from complete programs. By seeing statements *in situ*, he gains understanding not only of what each type of statement does, but also of how statements work together. By being exposed to statements in their natural environment, information about syntax and semantics is reinforced by example, and the student is discouraged from later using language constructs in misconceived ways. Hand tracing ("desk checking") provides further reinforcement. The very isolation of program reading from other activities is beneficial, in that the student cannot divert his attention from the programs. (We suspect many students only skim programs in textbooks.) Use of tools such as comprehension tests and class discussions of programs should increase the student's reading comprehension skills.

From reading, the student can gain some appreciation of the subtle factors that make programming an art; he learns idiom and style. The student can begin to see clearly that a program is not merely a translation of an algorithm. Rather, a program contains redundancies introduced for people--comments and indentation--as well as code designed to express to the compiler the assumptions upon which the algorithm was based, to circumvent limitations of the language, and so forth. These considerations are generally not part of the problem being solved, nor can they be derived from the problem; they are an aspect of programming *per se*.

When the student is ready to begin writing, he does so in a familiar and supportive environment. By being limited at first to program modifications, he can practice on a modest scale, yet still work on programs complex enough to be intrinsically interesting. By proper choice of examples, the student can be shown how small changes can have large effects on program behavior. Testing and debugging are more likely to seem manageable in this limited context. There is little reason for the student to thrash about or "hack" at a problem.

In the forth and final phase of the course, the student designs and writes original programs. By this time, he is quite sophisticated, having a firm grasp on the syntax, semantics, and most especially the pragmatics of the programming language. He has strong intuitions about how programs should behave, how they should be organized, and how they should be documented. He is ready to focus on techniques for outlining and developing programs. By the time the course ends, he may have written fewer programs than he might have in a more traditional course. We confidently predict, however, that he will be well-equipped to sharpen his programming skills, and will be relatively free of debilitating misconceptions. He is, in fact, ready to "take off" as a programmer.

SPECIAL REQUIREMENTS

Not surprisingly, a teaching method as unconventional as the one we propose requires its own set of special materials. To begin with, traditional textbooks are inappropriate. They do not support the first phases of the course and are too long to act as language references. A simple language manual can be used, or the students can actually construct their own references from class notes. In the latter case, the instructor should prepare an outline for the organization of the reference-to-be; no text is necessary.

The most conspicuous need for course materials is for a large collection of example (and exemplary!) programs. Some of these programs will only be run, some will only be read, but others must be available in both source and compiled form. At least some of the source listings should be compiler output. A certain number of source programs need to be available to students on-line and must be susceptible to realistic modification. It is important that the style of programs offered for serious study be of high quality and be consistent; students are expected to emulate that style.

Since the fundamental material of the course is programs of all sizes, this material must be distributed, rather than copied from a board or screen. Distribution of photocopies is possible, but having students run jobs to obtain copies of reading assignments is probably easier. Because lectures are high-volume presentations, use of a projector or video system is indicated.

POSTSCRIPT

The authors' suggestion for a more thoroughgoing analytic approach to the first programming course is based on many years of programming and teaching experience. We acknowledge, however, that our method is untried and that many of the assumptions upon which it is based have not been experimentally verified. We are now developing in greater detail the specific strategies and materials--the many goals, programs, exercises, etc.--needed to teach a course such as the one we have described. Also, we are outlining the criteria by which the degree of success of this approach may be judged when it is used in parallel with a more conventional approach. We encourage careful and controlled experimentation with these ideas by others. Additionally, we believe that our proposed approach to teaching suggests a wide range of questions for experimental computer science. (To what degree, for example, does program reading ability affect programmer performance?) It is important that these questions too be investigated.

REFERENCES

1. ACM Curriculum Committee on Computer Science. Curriculum 68: recommendations for academic programs in computer science. *Comm. ACM 11*, 3 (Mar. 1968), 151-197.
2. ACM Curriculum Committee on Computer Science. Curriculum '78: recommendations for the undergraduate program in computer science. *Comm. ACM 22*, 3 (Mar. 1979), 147-166.
3. Carberry, M. S., *et al.*, *Foundations of Computer Science*. Computer Science Press, Potomac, Md., 1979.
4. Chirlian, P. M., *Pascal*. Matrix Publishers, Portland, Ore., 1980.
5. Cress, P., Dirkson, P. and Graham, J. W., *FORTRAN IV with WATFOR and WATFIV*. Prentice-Hall, New York, 1970.
6. Deimel, L. E. and Pozefsky, M., Requirements for student programs in the undergraduate computer science curriculum: how much is enough? *SIGCSE Bulletin 11*, 1 (Feb. 1979), 14-17.
7. Dershem, H. L., A modular introductory computer science course. *SIGCSE Bulletin 13*, 1 (Feb. 1981), 177-181.
8. Dyck, V. A., Lawson, J. D., and Smith, J. A., *Introduction to Computing: Structured Problem Solving Using WATFIV-S*. Reston, Reston, Va., 1979.
9. Fisher, P., Hankley, W., and Wallentine, V., Separation of introductory programming and language instruction. *SIGCSE Bulletin 5*, 1 (Feb. 1973), 9-14.
10. Katzan, H., *Introduction to Computer Science*. Petrocelli/Charter, New York, 1975.

11. Kernighan, B. W. and Plauger, P. J., *The Elements of Programming Style,* 2nd ed. McGraw-Hill, New York, 1978.
12. Pattis, R. E., *Karel the Robot: A Gentle Introduction to the Art of Programming.* John Wiley, New York, 1981.
13. Prather, R. E. and Schlesinger, J. D., A lecture/laboratory approach to the first course in programming. *SIGCSE Bulletin 10,* 1 (Feb. 1978), 115-118.
14. Schneider, G. M., The introductory programming course in computer science--ten principles. *SIGCSE Bulletin 10,* 1 (Feb. 1978), 107-114.
15. Teague, R., *FORTRAN: A Discovery Approach.* Canfield Press, San Francisco, 1974.
16. Weinberg, G. M., *The Psychology of Computer Programming.* Van Nostrand Reinhold, New York, 1971.
17. Wulf, W. A., *et al.*, *Fundamental Structures of Computer Science.* Addison-Wesley, Reading, Mass., 1981.

11. Kernighan, B. W. and Plauger, P. J., *The Elements of Programming Style,* 2nd ed. McGraw-Hill, New York, 1978.
12. Pattis, R. E., *Karel the Robot: A Gentle Introduction to the Art of Programming.* John Wiley, New York, 1981.
13. Prather, R. E. and Schlesinger, J. D., A lecture/laboratory approach to the first course in programming. *SIGCSE Bulletin 10,* 1 (Feb. 1978), 115-118.
14. Schneider, G. M., The introductory programming course in computer science--ten principles. *SIGCSE Bulletin 10,* 1 (Feb. 1978), 107-114.
15. Teague, R., *FORTRAN: A Discovery Approach.* Canfield Press, San Francisco, 1974.
16. Weinberg, G. M., *The Psychology of Computer Programming.* Van Nostrand Reinhold, New York, 1971.
17. Wulf, W. A., *et al., Fundamental Structures of Computer Science.* Addison-Wesley, Reading, Mass., 1981.

A LIBERAL ARTS
COMPUTER LITERACY
COURSE
A PRELIMINARY REPORT

Stanley P. Franklin
Memphis State
University

A senior/graduate level course in computer literacy (broadly construed) for liberal arts students is currently being developed in the Department of Mathematical Sciences at Memphis State by Charles W. Brandon, III, Edward T. Ordman, Austin Smith, and myself. It's designed as a three semester-hour credit course with a one credit-hour lab which meets for three hours a week. A description of the course and its goals is included as Attachment I. The course, as originally conceived, is quite broadly based. Major areas to be covered in the lectures are hardware, systems, applications, organizational impact, and societal impact. Attachment II contains an outline of topics to be touched on in the lectures. The lab will be devoted primarily to a brief introduction to Basic programming and to the use of applications and communications packages. Attachment 3 contains a list of lab topics. Finally, Attachment IV offers an outline of intended methods of student evaluation.

Outside funds were raised for a microcomputer lab containing fifteen stations, each supplied with an IBM Personal Computer (64k of memory, one minidisk drive, and a monochrome monitor), an Epson MX-80 dot-matrix printer, and a Radio Shack direct-connect modem. With two students to a station, the lab accommodates thirty students. Renovations and installation were completed early in January of 1982.

One section of the computer literacy course is being offered for the first time during the spring semester of 1982. We intend to revise the course during the Summer of 1982 in the light of this first semester's experience, and to present the revised version to several sections in the fall semester of 1982. During the oral presentation of this paper I hope to report on the successes and failures of this first attempt.

ATTACHMENT I
Description of a Course in Computer Literacy

This course is designed for the liberal arts major with a minimum prerequisite of a single college level course in mathematics. The course has two goals:

1. Enable the student to become comfortable with the hands-on use of modern, office type computing equipment. The student will become accustomed to utilizing the equipment both as a stand-alone computer and as a terminal, communicating with some of the major data banks and the University computer center.
2. Expand the student's mind and level of awareness in the best sense of the liberal arts tradition. This course is a broad survey of the full spectrum of computing in our modern society and is definitely not a science or mathematics course. The viewpoint of the course values the cultural, artistic, societal, organizational, and psychological aspects equally with the engineering, mathematical, theoretical, and scientific aspects.

The course will achieve the first goal largely by intensive exposure to the use of modern microcomputers in a non-threatening setting. The course is structured to provide a three hour per week laboratory with one hour of academic credit. Over the fifteen weeks of the course, the student will have a minimum of 45 hours of hands-on experience. Students will be paired, two to a computer. The microcomputer will have a substantial capability for computing in its own right. The computer will also be operated as a terminal on the University's main computer and on a number of the large public access data banks. The laboratory assistants will be carefully selected and trained to offer a sympathetic and helpful attitude to the novice computer user.

This first goal of becoming comfortable

with the computer is of great practical benefit to the student. James Martin, a former IBM researcher and lecturer, estimates that by 1990 some 70% of the work force will be working with the computer in some fashion. His book, The Wired Society, predicts a world where home and office are interconnected on a nation wide basis with high speed communications links. The student who is comfortable with the use of the computer will have a distinct practical advantage in the work and professional environment over those persons who do not know how to take advantage of the computer as a tool. When a new computer system is introduced, graduates of this course will likely be among the first volunteers to learn how to use it rather than hanging back out of fear of the unfamiliar. At home too, the graduate of this course will have an advantage. Whether in his finances, the education of his children, or in other ways, the student will be prepared to benefit most quickly from new, low-cost computers in the home.

The course will achieve the second goal primarily through a stimulating, three hour per week lecture course with three hours of academic credit. The lectures will be newly developed and will cover the full spectrum of computing in our world. The attached outline of topics in computer literacy will give the reader some idea of the breadth of this course. Many issues seldom, or in some cases, never dealt with in conventional computer courses, will be touched upon in some depth.

Computing today is an immature discipline which has yet to achieve a complete sense of its own identity. The spectacular advances in computing technology tend to obscure the irregular development of computing as a professional discipline. The situation resembles the state of engineering in the early part of this century. Computer programs are still largely the products of engineers. The immature state of the discipline is reflected in the immaturity of the academic curricula available to students of computing. Most computer programmers today still learn the major aspects of their work in an apprentice-like fashion in the work environment. For the non-professional, the world of computing appears even more mystical. Yet with increasing frequency, all of us as citizens, parents, professionals, workers, and thinking human beings are faced with decisions regarding computers and their use in our world.

Should the IRS and Social Security computer data banks be allowed to cross reference each other? What about banks, eliminating paper checks in favor of all electronic records of our financial transactions? Who should be allowed to access our medical and insurance records which are stored in computer data banks? What kind of advantage will a child have in entering college who has been exposed from age 5 on up to a computer in the home? How can we detect computer criminals? Is a computer-produced musical score for a major motion picture a true work of art? What will happen to musicians if more and more studios turn to this tool? Can computers really think? What do we mean by "thinking" anyway? These and hundreds of other questions and issues are beginning to confront us as the first citizens of a computerized society. Our only hope for wise decisions in these matters is the creation of a populace broadly educated in computing in the context of the liberal arts and sciences.

This course will particularly emphasize those aspects of computing which are somewhat independent of particular technology and therefore will have a more lasting value to the student. The course will tend to treat the computer itself as more of a "black box" though it will not fail to expose the students to some of the latest "miracles" in technology. It will tend to view computing from the perspective of whole systems, organizations, and society. This "meta-level" point of view is not widely perceived in the computing community and will be something of an innovation in computer education.

The laboratory and lecture sections will be carefully tied together in content and pacing so as to reinforce each other to a substantial degree. By utilizing an up-to-date microcomputer laboratory, the students will have first hand experience with some of the topics, problems, and issues raised in the lectures. In particular, this is why the use of some of the national public access data banks will play a significant role in the laboratory. The students will have the opportunity to read from the electronic edition of "The Washington Post." They will use the computer as a word processor to prepare their required reports. They will prepare a simple home budget using the computer as a financial modeling tool. They will send and receive electronic mail to each other. They will search one or more of the large data banks for references as they research their term papers. The students will experience new forms of play as they try their hands at competing against the computer in chess or other games. In these and other ways the students will gain both a practical and theoretical understanding of computing.

In addition to the lecture and laboratory activities, the students will be encouraged to read from reference material in an augmented library collection of the latest in magazines and classic books on computing topics.

The application of computer technology is working a revolution in our society. Every effort will be made in this course to enable the students to become educated and reflective citizens of the Computer Age. By its broad intellectual scope the course will further invite the student to the exciting intellectual adventure made possible by computer technology.

ATTACHMENT II

An Outline of Topics in Computer Literacy

The Hardware Level: Machine Architecture; Logical Machine Design; Boolean Algebra; Circuitry (Electronic and Photonic) Integrated Circuit Fabriaction Technology; Mass Storage Device Technology.

The Theoretical Level: Theory of Programming Languages; Artificial Intelligence; Automata Theory; Theory of Algorithms; Numerical Analysis; Control Theory; Database Theory.

The System Level: Instruction Set - Native Languages; Interpreters, Compilers, and Assemblers; Operating Systems, Logical Machines; Virtual Machines; Database Management Systems; Communications Interfaces.

The Applications Level - Whole Systems: Computers + Networks + Terminals; Software as Engineering; Systems Design Methodology; Project Control and Management; Systems Integration; System to System Compatibility; Scientific Computing.

The Organizational Level: The Life Cycle of Data Processing in Organizations; Impact of Data Processing on Organizational Structure; Security (Physical and Data); Economics of Computing.

The Societal Level: New Tools for Knowledge Workers; Diffusion of Availability of Access to Information; Psychological Impact ("The Hacker Mentality"); New Careers (Information Specialists); New Business Opportunities (Information Providers); New Forms of Play (Intercontinental Chess); The Privacy Issue (EFT, IRS, Personal Data); Displacement of Workers; The Computer Dependent Society; Cultural & Artistic Aspects (Music Synthesizers, Graphics), Computer Crime (Computer Capers); International Information Imperialism; Shifts of Consciousness.

ATTACHMENT III

Laboratory Course Structure

Topic	# Labs/ # Hours
Hardware Logic Demo	2 / 6
Computer Familiarization Games	1 / 3
Word Processing	2 / 6
VisiCalc	2 / 6
Data Base Programming	1 / 3
On-Line Public Data Banks	3 / 9
SPARE	1 / 3
Totals	15 / 45

ATTACHMENT IV

Computer Literacy - Basis for Student Evaluation

A. Hour examinations - multiple choice several forms
1. First exam - hardware and theoretical levels
2. Second exam - systems and applications levels
3. Third exam - organizational and societal level

B. Final examination - comprehensive, multiple choice, several forms

C. Term project - some approved course topic researched to some depth
1. Report to be written on computer word processor
2. Some data bank accessed on-line for information
3. List of suggested topics available by midterm
4. Students free to choose own topic subject to approval

D. Lab assignments - students in pairs on microcomputer
1. A simple program to write
2. A more complex program to modify
3. Use of applications program such as:
   - (a) word processing
   - (b) electronic spreadsheet
   - (c) terminal program
   - (d) data base management program
   - (e) statistical package
   - (f) general ledger
   - (g) mailing list
4. Access of several on-line data banks
5. Use of computer as control device

SMALL ONE USER SYSTEMS VERSUS TIMESHARING SYSTEMS IN THE INTRODUCTORY COURSE A CONTROLLED EXPERIMENT

by

Charles M. Shub
University of Vermont
Burlington, Vermont 05405

ABSTRACT

Data quantifying the effect of different hardware configurations on the teaching of an introductory programming course is presented. The experiment and control groups are described. The evaluation model is detailed. The results of the evaluation are presented. Conclusions are drawn.

INTRODUCTION

In the fall of 1981, the Computer Science Program at the University of Vermont (UVM) used a new methodology to teach the introductory service course. Approximately 250 students, the control group, had timesharing access to a Harris 500 computer, the same access that UVM had used for a number of years in this course. Another group of about the same size, the experimental group, had computing access through stand alone systems, namely TERAK 8510 computers, a PDP 11/02 based machine. To our knowledge, this is the first reported instance of such a dichotomy of hardware used for two sections of an introductory service course. The details of why this configuration was used have been published elsewhere, [SHUB82] and will not be discussed in great detail. Of relevance to the configuration selection is the work of Bowles [BOWL78] who has successfully used the TERAKs in an introductory course at the University of California at San Diego, the extensive program development software available for the TERAK as reported recently in The Communications of the ACM, [TEIT81] the cost data presented by Goldbogen, [GOLD81] and the common practices around the country with respect to language selection. [HILL80] The mode of teaching was that of large lecture sections meeting for two hours a week, and then discussion, recitation or lab sections with approximately 24 students meeting for an additional two hours. These sections were run by senior or graduate student Computer Science majors. The two lecture sections were taught by the same individual. A common text [LIPS78] was used to teach Fortran according to the 1966 standard extended by quote fields and format free I/O. Both sections were assigned the same programs. Common hour examinations were given. The Harris system used a simple line editor for entering data and programs, while the TERAK systems used a simple screen editor somewhat similar to the UCSD Pascal editor. [BOWL78]

THE TECHNICAL EFFECTIVENESS

In order to measure the relative technical effectiveness of the two hardware alternatives, a sixteen point questionnaire was administered to all students on two separate occasions. The first time was approximately midway through the semester, and the second time was during the last week of classes. This questionnaire was used to collect data regarding the amount of computer usage, the utility of locally prepared handouts, the utility of system error messages, system hardware reliability, system accessibility, contention, ease of getting hard copy, and availability of tutorial help.

The specific questionnaire was modified slightly from an earlier version developed and tested in the summer of 1981. The modifications removed some minor ambiguities. The committee which made the changes included the author, the original developer of the questionnaire who is employed by the UVM Instructional Development Center, and a statistician on the staff of the Academic Computing Center. The questionnaire, in its earlier version, was tested and validated by a pilot group of 48 students who participated in a test run of the course during a six week summer session in 1981. [SHUB82]

The primary motivation for the study of relative effectiveness was to assist the central administration in long range planning for acquisition of additional equipment to be used for the anticipated enrollment increases in Computer Science in the future.

DATA VALIDATION

At this point we had eight survey instruments:

H6: Harris Section after 6 weeks
H12: Harris section after 12 weeks
T6: TERAK Section after 6 weeks
T12: TERAK Section after 12 weeks
HM: Harris Section Mid semester Summer pilot project
HF: Harris Section Final Summer pilot project
TM: TERAK Section Mid semester Summer pilot project
TF: TERAK Section Final Summer pilot project.

Validation tests for internal consistency were made on the first four data sets. Similar tests could not be made for the latter four sets because the number of students was too small. These tests involved breaking up the data into two groups in at least four different arbitrary fashions (by section, by order of input, by alternation, and by batched alternation) and testing for consistency of the two subsets by applying the Student T test for equality of means. No statistically significant differences were found within any of the four data sets. We concluded that the data was internally consistent for each of the four data sets gathered during the fall of 1981.

DATA ANALYSIS

Student T tests were run on the following eight pairs of data sets for the reasons indicated:

A. H6 to H12. Run to detect trends in the Harris section

B. T6 to T12. Run to detect trends in the TERAK section

C. H6 to T6. Run to detect differences between the sections at mid semester

D. H12 to T12. Run to detect differences between the sections at the end of the semester

E. H6 to HM. Run to see the effects of class size and equipment contention on the Harris Computer at mid semester

F. H12 to HF. Run to see the effects of class size and equipment contention on the Harris Computer at the end of the semester

G. T6 to TM. Run to see the effects of class size and equipment contention on the TERAK Computer at mid semester

H. T12 to TF. Run to see the effects of class size and equipment contention on the TERAK Computer at the end of the semester

RESULTS

A. Harris Section Trends:

1. Students had less trouble understanding the error messages at the end of the semester. (T=3.10)
2. There was less time spent waiting for a vacant terminal at the end of the semester. (T=2.84)
3. The terminals were perceived as more accessible at the end of the semester. (T=3.11)
4. The text editor was perceived as less satisfactory at the end of the semester. (T=2.69)
5. The system error messages were less satisfactory at the end of the semester. (T=4.79)

B. TERAK Section Trends:

1. There were fewer terminal sessions at the end of the

semester. (T=5.75)

2. Students had less trouble understanding the error messages at the end of the semester. (T=5.62)
3. There was less time spent waiting for a vacant terminal at the end of the semester. (T=7.47)
4. The terminals were perceived as more accessible at the end of the semester. (T=5.38)
5. Students felt tutorial help was more available at the end of the semester. (T=3.35)

C. Mid Semester Cross Comparison:

1. Harris students visited the terminals more often. (T=3.87)
2. Harris students had to stop work due to equipment failure more often. (T=4.39)
3. Harris students had shorter waiting periods for available terminals. (T=2.55)
4. Harris students were more satisfied with the locally prepared handouts. (T=3.38)
5. Harris students had to wait longer for hard copy. (T=5.04)
6. Harris students needed less help with the text editor. (T=3.09)
7. Harris students found the terminals more accessible. (T=2.81)
8. Harris students found the terminal locations more convenient. (T=4.46)
9. Harris students found the system error messages more confusing. (T=2.69)

D. End of Semester Cross Comparison:

1. TERAK students made fewer trips to the terminals. (T=4.51)
2. TERAK students had shorter waiting periods for available terminals. (T=2.25)
3. TERAK students were getting hard copy faster. (T=6.69)
4. TERAK students found the terminals less accessible. (T=2.30)
5. TERAK students found the terminal locations less convenient. (T=3.36)
6. TERAK students found the text editor more convenient to use. (T=2.59)
7. TERAK students found the error messages less confusing. (T=7.46)

E. Mid Semester Summer Versus Fall Harris Computer:

1. Summer students waited less for terminals. (T=3.85)
2. Summer students found the terminals more accessible. (T=6.57)
3. Summer students found the handouts more accurate. (T=2.97)
4. Summer students found the equipment more reliable. (T=3.43)
5. Summer students found the tutorial help more available. (T=3.37)
6. Summer students found the terminal location more convenient. (T=4.16)
7. Summer students found the text editor easier to use. (T=4.46)
8. Summer students found the error messages clearer. (T=3.49)

F. End Semester Summer Versus Fall Harris Computer:

1. Summer students waited less for terminals. (T=3.46)
2. Summer students felt the terminals were more accessible. (T=3.46)
3. Summer students felt the system was less reliable. (T=3.07)

4. Summer students felt there was more tutorial help available. (T=2.83)

5. Summer students were more pleased with the location. (T=3.45)

6. Summer students found the text editor easier to use. (T=2.81)

7. Summer students found the error messages clearer. (T=5.30)

G. Mid semester Summer Versus Fall TERAK Computer:

1. Summer students had more trouble understanding error messages. (T=3.16)

2. Summer students waited less for terminals to become available. (T=4.83)

3. Summer students felt the handouts were less accurate. (T=4.31)

4. Summer students felt tutorial help was more available. (T=2.97)

5. Summer students liked the terminal location better. (T=5.67)

6. Summer students found the error messages clearer.(T=2.79)

H. End semester Summer Versus Fall TERAK Computer:

1. Summer students had more trouble understanding the error messages. (T=4.02)

2. Summer students waited less for terminals to become available. (T=3.94)

3. Summer students waited less for hard copy. (T=2.80)

4. Summer students felt the terminals were more accessible. (T=8.02)

5. Summer students found more errors in the handouts. (T=3.84)

6. Summer students felt the equipment was more reliable. (T=3.19)

7. Summer students felt tutorial help was more available. (T=4.60)

8. Summer students liked the terminal location better. (T=5.08)

CONCLUSIONS

The conclusions which we draw from the above data fall into four general categories including student learning, the effects of class size, handout development, and cross comparison of equipment.

Learning Trends

The trend data is, in general, not at all surprising. We expect that the students will become more knowledgeable of the system as the semester progresses and that they will thus become more familiar with the use of the text editor and the meaning of the error messages. This was predicted by the summer results reported earlier. [SHUB82] What is interesting is that the students apparently learned to deal effectively with the contention problems. We see this manifested in several fashions. Although the total amount of time spent at a terminal did not significantly change in either fall section, we note shorter waiting periods for terminals and a perceived increase in terminal availability. We presume that this is because the students learned they could get better access in the evenings and on weekends and thus made use of those time periods to maximize their productivity.

On the TERAKS, we also note a tendency toward spending longer periods at the terminals. We presume that having gotten to the terminal site, the student will work to completion rather than make several partial attempts. This item is addressed further in the subsection on equipment comparison.

Effect of Class Size

The effects of class size were much greater than we expected. The results of the last four comparison runs bear this out. We had expected to see the effects of contention in terms of waiting times and the accessibility of hard copy. However, we did not anticipate the other side effects of class size described below.

In general, the smaller classes were much more pleased in all areas than the corresponding larger classes. We attribute this to the personal attention the students in the summer sections received. During the summer the same teaching assistant was always available for tutorial help no matter what day of the week or time of day the student needed help. Moreover, the summer teaching assistants knew all the students by name. In the fall, a student might receive help from any one of a dozen assistants, and quite possibly never the same person twice. Moreover, the specifics of suggestions might differ from one assistant to another, thus leaving the student confused as to which approach to follow.

The greater ease of use of the text editors in the summer can be attributed to the ability of the assistants to offer one-on-one instruction and assistance. In addition, the denser pace of the summer section (6 weeks versus 14 weeks) meant that the students would be working on the terminals every day. Thus it became part of the daily routine to attend class and then go to the terminal room. As such, going to the terminal room was not perceived as making a special trip to an inconvenient location.

Handout Development

One function of the technical effectiveness survey was to serve as a critique of locally prepared handouts. In the preliminary summer session, students were highly critical of the first cut at a handout for the TERAK systems. As a result of that experience, the TERAK handout was substantially revised prior to the fall session. The last two cross comparisons show that the Fall students were much happier with the handout than the summer students. While the mid semester comparison in the fall still showed some dissatisfaction with the TERAK handouts, the dissatisfaction disappeared as a significant indicator by the end of the semester. The TERAK handout is now going through its second revision, and is expected to be as accurate and easy to use as the Harris handout.

Equipment Comparisons

The two cross comparison runs, one at mid semester and the other at the end of the semester provide the basis for most of these conclusions. Those statistics are augmented by the analysis of the summer data which also pointed out similar concerns.

One major difference between the two pieces of equipment is the difference in the style of the text editors. We have concluded that the Harris text editor, a simple line oriented editor, has a lower learning threshold than the screen editor of the TERAK system. This conclusion is based upon the Harris students having significantly less trouble with the text editor than the TERAK students at mid semester. We also conclude that as programs grow larger, and the threshold appears to be about a screen full or 24 lines, that the screen editor begins to show its superiority. The screen editor allows one to easily focus on an area of code and debug that section without having to return to the beginning. The renumbering of subsequent lines by the Harris editor after the student either inserts or deletes lines seemed to cause problems. This conclusion is supported by the end of semester comparisons in which the TERAK students liked the editor better. This conclusion also jells with the author's personal feelings, which is comforting.

The data also supports the conclusion that the stand-alone system is more convenient from a student's point of view. There are two items that indicate this. First, the shorter delays in getting hard copy are clearly shown in the data. Waiting for hard copy is a substantial time waster in the batched hard copy mode of the Harris. Moreover, while the Harris system is available to users for about 150 hours a week, students can only pick up hard copy during the 8:00 AM to Midnight periods. However, on the TERAKS, the two printers proved sufficient to provide hard copy with sufficient rapidity. The other area where we see more utility of the stand-alone systems is the area of reliability. On the Harris systems, the system is either up or down. Either everybody can use it or nobody can. The issue of a broken terminal does not enter into consideration because the Harris is port capacity limited and there are always at least as many operable terminals as ports. In the case of the TERAKs, however, the failure of one machine does not stop everybody from working, but rather merely increases the contention for the remaining machines. Thus in terms of reliability, we strongly feel that the stand-alone system provides a substantially more reliable and effective computing environment for the students.

In terms of the waiting line and accessibility data, the results appear inconclusive insofar as at mid semester the Harris was better and at the end of the semester the TERAK was better. However, given that the Harris system also supported another class, one might conclude that the lighter load on the Harris from the other class made the Harris appear to perform better than expected earlier in the semester.

Finally, in terms of location convenience and accessibility, the the data says the Harris System wins hands down. This is because The TERAK systems are all located in the same room of a single building and are accessible only 96 hours a week while the Harris terminals are available for about 150 hours a week. Moreover, the computing center provides terminals to access the Harris at a variety of locations around campus including the library and several dormitories.

BIBLIOGRAPHY

[BOWL78] Bowles, Kenneth L., "A CS1 Course Based on Stand-alone Micro Computers", Proceedings of the SIGCSE/CSA Technical Symposium on Computer Science Education, Detroit, Michigan, February, 1978, pages 563-573.

[GOLD81] Goldbogen, Goef and Williams, C H., "The Feasibility of Persona Computers Versus a Minicomputer for a Computer Science Program" SIGCSE bulletin, Volume 13, No. 3, September, 1981, pages 2-4.

[HILL80] Hill, David B., "Programming Languages for Service Courses and Courses for Computer Scienc Majors", SIGCSE Bulletin, Volum 12, No. 4, December, 1980, pages 43-45.

[LIPS78] Lipschutz, Seymour and Poe, Arthur, "Programming With FORTRAN", McGraw Hill, 1978.

[SHUB82] Shub, Charles M., "Does the Computer System Make a Difference in the Effectiveness of the Introductory Service Course?", Proceedings of the 13th SIGCSE Symposium on Computer Science Education, Indianapolis, Indiana, February, 1982.

[TEIT81] Teitelbaum, Tim and Reps, Thomas, "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment", Communications of the A.C.M., Volume 24, No. 9, September, 1981, pages 563-573.

Engineering

Low Level Computer-Assisted Design on an APPLE II With Pascal

Ralph M. Deal
Chemistry Dept.
Kalamazoo College
Kalamazoo, MI 49007
(616) 383-8452

THE TASK

Faculty members at small colleges do not have access to the kinds of research support facilities available at large universities, industrial, and national laboratories. One of these missing facilities is a draftsperson to produce engineering diagrams for machinists to use in producing special equipment. Faced with this problem when I needed to produce accurate diagrams of a special cell for my research in thermal energy storage, I decided to use the department's APPLE II with language card and a printer to produce appropriate "drawings". I was fortunate enough to find a skilled mechanic in my area who could produce my cell but I wanted to minimize the chances for errors in using my meagre research funds to pay the machining costs.

IMPLEMENTATION OF A TWO-DIMENSIONAL CAD SYSTEM

Having had considerable experience in producing graphics via APPLE Pascal's "Turtlegraphics" (a relative polar graphics command set developed in MIT's LOGO language), I decided I needed a fast flexible drawing procedure that would act as an interpreter with immediate feedback and error-correction capability. In 1979, I worked with a similar but less interactive design system at the Oak Ridge National Labs and so I knew what capabilities I wanted the system to have. UCSD Pascal programs are compiled via a tediously slow procedure which hardly suited my purposes. Therefore, I wrote a menu-driven interpreter whose basic commands were turtlegraphic commands or suitable combinations thereof. By using the screencolor of "reverse" in drawing figures on the screen, I could redraw a misdrawn figure with the same parameters to erase that figure.

The current set of commands includes B(ox, C(ircle, A(rc, L(ine, T(urn, F(orward, M(oveto, D(ash, W(here[am I], G(rid, and teX(t. As the number of commands in my set grow, I needed a pair of prompt lines at the bottom of the screen to remind me of my options. By using mode #6 in writing this pair of prompt lines, I could simply rewrite the lines to remove them, preserving any figure lines in the same space.

Rather than use some qualitative interactive graphic input system such as paddles and a flashing cursor, I decided that for engineering drawings, I wanted the accuracy of numerical input, where I had to make explicit allowance for the finite resolution of the screen. As a result, the user is prompted for appropriate screen coordinates for each command.

Some care had to be exercised in adjusting the method for producing some figures to allow for the low resolution of the screen (280 x 191 pixels). For example, in drawing a circle, the actual number of degrees turned per segment had to be increased as the radius of the circle was decreased so that the truncation of the calculation of how far to move did not produce too much distortion.

After a reasonably satisfactory diagram has been produced, I use a hardcopy of the list of commands that I entered, clean out any erasure commands, and reenter these commands with a MAKE command. This Apple Pascal feature allows you to store a set of keystrokes in a file which you can later execute to reproduce the same input sequence. By exercising some care, this EXEC file can be edited to clean up text strings, alter coordinates, even enter new commands. The care is necessary because of the peculiar form of EXEC files; for example, you cannot enter new lines and expect them to work. However, you can copy existing lines in the EXEC file and modify them. While this may seem quite awkward, it is straightforward for anyone conversant with the Pascal editor. Each new EXEC file is tested by having it draw a new diagram until a diagram is ready to be submitted to the machinist. The EXEC file then becomes a file form of the image produced and can be used to reproduce the diagram, perhaps in slightly altered form, anytime later. An example of a simple alteration that I found helpful is to do all test runs in a "normal" image size on the printer and then alter the file to ask for the "expanded" form when producing the final copy.

Just before labelling the diagram at the end of a drawing sequence, I use the G(rid command which allows me to produce a grid of + signs on centers spaced by intervals familiar to the machinist. My original grid pattern used dashed lines with no correlation between the intersection of grid lines. This produced a confusing pattern and so the dashes are calculated to occur at grid line intersections. This grid is written in an absolute mode so that the grid will appear whether a figure line happens to coincide. Of course, one could use color to distinguish grid lines and to distinguish parts of the apparatus being drawn but since my output medium was a IDS 460G printer, color was not an option. While I could have drawn the figures in color on a color monitor/TV and photographed this display, the delay time and cost in producing the large prints would be prohibitive; another disadvantage would be losing the accurate spacing over the entire grid obtained with the printer. In any case, machinists are used to monochrome blueprints.

After the grid is produced, a L(abel command allows me to place upper/lower case strings on the diagram as headings and labels. While I could have added arrows from the labels to specific parts of the apparatus drawing, using paddles and a blinking cursor or paddles and a "rubber-band", this seemed unnecessary since it is easy to add such arrows by hand.

Similarly, I did not implement a diagonal hatching scheme to identify certain areas because of the relative ease of doing so by hand and the awkwardness of the algorithm for doing so automatically; it would require keeping all previously drawn lines in memory and determining closed areas from that information. I have worked with such an algorithm with a CAD system referred to earlier called MAPPER at the Oak Ridge National Labs and had some difficulty in achieving the effect I wanted. (MAPPER was developed at the Los Alamos National Laboratory and was adapted to the set of Oak Ridge computers by the ORNL computer division staff).

Here are two examples of output from the 2D program. Both are of the cell I had made by the machinist. Figure 1 is an endview of the cell showing how the window is held on by an external frame and where the O-ring slot is cut into the body of the cell. Several copies of this diagram were made and on each one a specific part was emphasized by diagonal lines. These separate diagrams featured the end window itself, the cell body, and the frame holding the end window in place.

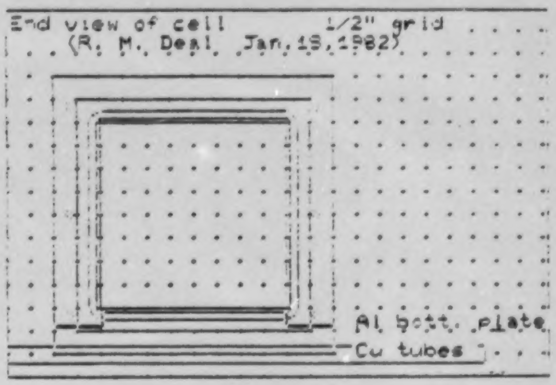


Figure 1

Figure 2 is a view of the side of the bottom plate of the cell with its grooves for fitting eight copper tubes. Note the change in scale for this diagram. One should produce a labelled scale for the machinist to use when taking dimensions off these diagrams, one for each different scale used. I used 12 pixels per grid unit in both cases because it happened to fit my problem well but 12 is probably a good choice in any case since it allows you to accurately fit 1/2, 1/3, 1/4, 1/6 of a grid unit in placing lines.

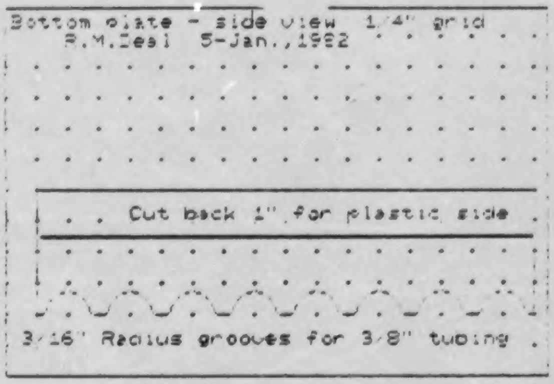

Bottom plate - side view
1/4" grid
R.M.Deal 5-Jan.,1982
Cut back 1" for plastic side
3/16" Radius grooves for 3/8" tubing

Figure 2

THREE-DIMENSIONAL COMPONENT

Since perspective drawings help most people understand better a three-dimensional object, I decided to augment my dimensionally accurate 2D diagramming system with some more qualitative perspective drawings which would ensure that the machinist and I were thinking of the same object. This was not difficult for me because of a 3D program I had already worked with, namely a little-publicized Pascal implementation of the excellent 3D graphics system developed by the Sublogic Company in Champaign, Illinois, best known for its top-ranked game using the graphics system, Flight Simulator. The Pascal version was not very well developed and seemed useless to me in the form I first received it but is now quite useful, especially in this kind of application. The task here is to translate the numerical values I had developed for the parts of the diagrams in two dimensions by adding a third dimension and feeding these coordinates to the 3D program. That program then allows me to use a joystick and two paddles to move around and zoom in on the image to give me a 3D picture of the apparatus that seems to show critical aspects of the apparatus well. By using the same coordinates developed in the 2D diagrams except for the third coordinate, I can be reasonably sure that the translation is dependable.

Figures 3-6 show some of the kinds of views obtainable with this program. The object is the same cell shown in Figures 1 and 2. Making my initial choice of height and pitch (the viewing angle) by keyboard and moving around the object by a joystick control and controlling the horizontal viewing angle and the extent of zoom by separate paddles, I had complete control over the image and could place myself actually within the cell wall as shown in Figure 6.

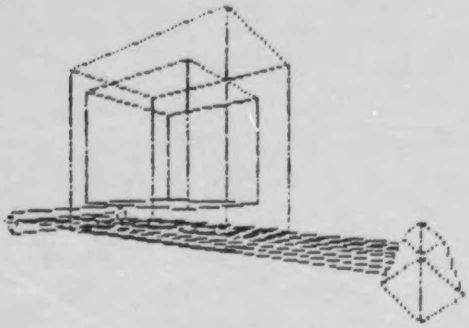


Figure 3

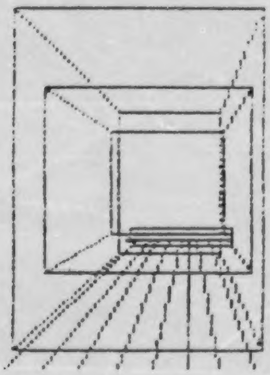


Figure 4

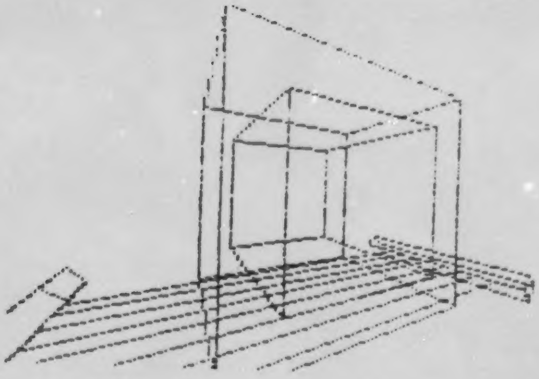


Figure 5

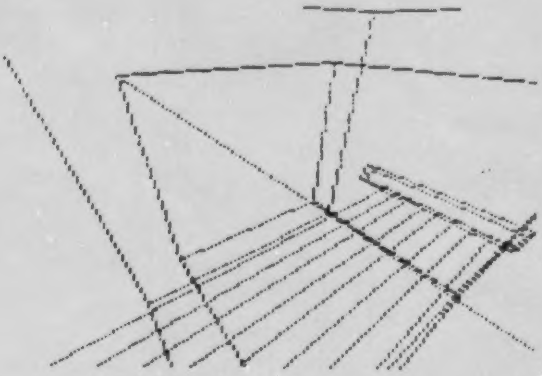


Figure 6

Unfortunately, in its present form, producing an image for the 3D display is too time consuming to serve the kind of fast turn around modification available on the 2D display system. With the use of EXEC files, the actual number of steps a user has to execute can be minimized but the sequence still must produce the file of coordinates, convert it to assembler form (via a program I call DEVELOP after a program serving a similar function in Sublogic's Basic version of the graphics package), assemble it, link it to the master Pascal program, a UNIT that protects the second graphics page, and the graphics code. Before the system can be used easily, there should be a program that produces an immediate 3D image as the user asks for simple shapes such as lines, circles, 2D, and 3D boxes, etc.

CONCLUSION

Based on my experience with the specific application used here, the system is useful. With it, I have been able to take a series of drawings to the machinist, make corrections based on his suggestions, enter those corrections on the diagram, and resubmit the diagrams. Once the system was developed, this cycle took very little time at the computer and hence there is some basis for calling this system a computer-assisted design system, despite its meager capacities compared to the rapidly developing dedicated CAD systems described in the current literature(1).

1. See, for example, the February 1982 issue of Datamation.

INTERACTIVE MICROCOMPUTERS WITH GRAPHICS PROVIDE TEACHING ASSISTANCE IN ENGINEERING AND TECHNOLOGY

William R. Moore and Robert L. Baxter
Dept. of Industrial & Engineering Technology
Western Kentucky University
Bowling Green, KY 42101
(502) 745-2461

INTRODUCTION

The electrical engineering technology program at Western Kentucky University is presently using APPLE II microcomputers in an interactive mode for: (1) testing student understanding of laboratory concepts, (2) providing students with individualized homework assignments, (3) providing specific help for students who need assistance, and (4) enabling students to increase their knowledge of electrical concepts at their own pace outside of the classroom.

The idea for our interactive laboratory reporting and testing system came about one day in 1978 during an informal faculty discussion centering on poor laboratory performance of many of our students -- both seniors as well as underclassmen. In particular, student operation of basic electrical laboratory equipment such as oscilloscopes was poor. Seniors, nearing graduation, couldn't properly synchronize a scope, didn't understand instrument and circuit loading concepts, and often had trouble connecting up all but the simplest circuits. Student reports of laboratory results were poor. The reports often showed no analysis of results, incorrect conclusions drawn, and occasionally evidence of student copying of reports. We felt that we needed more emphasis placed upon proper equipment operation, possibly through a one-on-one verbal exchange with each student. Ideally this exchange should question the student on his understanding of laboratory concepts, equipment operation, and the effects of equipment impedance upon the accuracy of results. Since it wasn't feasible because of time limitations to arrange this exchange between a faculty member and all the students enrolled in a course, a computer operating in an interactive mode seemed a logical choice.

A proposal was prepared and submitted to the National Science Foundation (NSF). This proposal was funded in 1980, and we were committed to getting the computer system up and running.

Funds for this project allowed us to purchase a video cassette player and video tapes. This equipment was to be used to provide quality instruction on equipment, its operation, and effects upon experimental results.

INTERACTIVE COMPUTER TESTING

The interactive computer testing system uses an APPLE II computer to test students enrolled in a sophomore electrical laboratory course. This course introduces students to diodes, transistors, and single-stage amplifiers. There are six tests on the computer to test students on their understanding of nine laboratory experiments performed during the semester. Students record their experimental data in a laboratory notebook for later reporting and entry into the computer.

Our computer system has four versions of each laboratory test. Each of the four versions are similar, but no questions are identical. The circuit component values used in the questions are all randomly generated. In this way, no two students take exactly the same test. Each of the students in each laboratory experimental group is assigned a different version of the laboratory test to further insure that laboratory partners take different tests.

The four versions of a laboratory test for a particular experiment are stored on a standard 5 1/2-inch floppy disk with a "greeting" program.

The APPLE II microcomputer has an auto-start feature which automatically boots the disk and runs the greeting program when the computer is first turned on. Our greeting program is used to say hello and provide additional instructions on the system operation. It also checks to see if the student has completed all required laboratory tests and is ready to take the selected test. If the student has satisfied all of the conditions, the correct version of the laboratory test is automatically loaded and run. There are 16 to 20 questions on each laboratory test.

The value of each test question for grading purposes can be independently set by the instructor as well as the number of times the question is to be repeated if missed. The value to be deducted for each incorrect response can also be set to any desired value. Each question can have any number of help statements. However, only one help statement will be given for each student response.

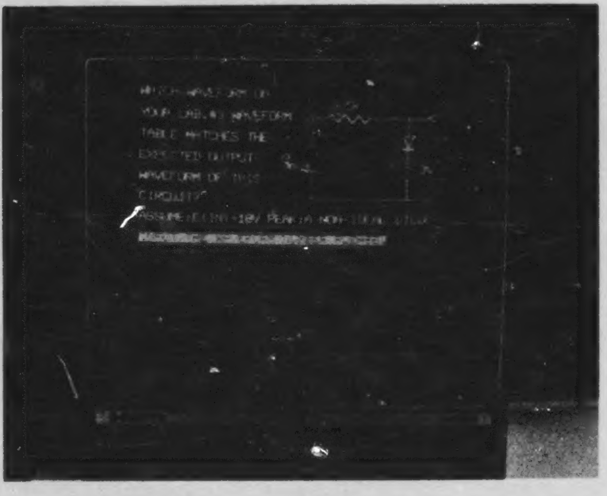


Figure 1. A laboratory test question on diode clippers showing the use of high-resolution graphics to draw the electrical schematic.

SYSTEM SECURITY

Considerable effort was spent in the design of the system to insure the security of the test questions. Non-printable control characters are imbedded in the program names to prevent anyone from loading a test and looking through it. In addition, the system is designed such that once a lab test is begun it cannot be stopped except via a normal routine which saves the student data. When the student next runs the lab test, he or she is returned to the same question from which he or she left. A student is allowed to stop the test twice without penalty. From then on, each exit will deduct ten points from the final score.

All student data is kept in a random access file. For security reasons, the student's laboratory number, question number, total score, percent score, and similar data are kept in certain bit locations of fixed-point variables. Therefore, even if a student had access to the student data disk, he would be unlikely to be able to read or change the data.

The system is designed so that it will run on an APPLE II microcomputer having either one or two disk drives. If there are two disk drives, the student data disk goes in one and the laboratory test disk in the other. For a one-disk system, the student begins with the laboratory test disk and is told when to insert the student data disk. The disks have to be changed twice unless the student has previously left the test -- then a third swap of the disks is required. There is only one student data disk but three copies of the test disks. Several students may take the same laboratory test simultaneously using the three APPLE's which we have available. They each have to wait until they can use the student data disk which is needed only to start the test and to finish or leave the test. This takes only a few seconds.

The instructor has two programs available: one for creating student files and the other for obtaining student grades and progress reports from the six laboratory tests. The file-manager program, besides creating files, can add or delete students from an existing file and modify any student data. The report program will report student results organized either by laboratory course number or by student name.

MODULAR PROGRAMS

Our laboratory testing system proved so successful that we decided to use many of the ideas and programming techniques developed there to enhance student learning in other courses. We decided to develop program modules which would demonstrate and provide practice with a series of individual electrical concepts, such as Mesh Analysis or Thevenin's Theorem. These programs would be combined on a floppy disk with a menu program which would allow us to save the number of times the program had been run as well as the user's name and score. In essence, we could create on a single 5 1/4-inch disk a complete tutorial package covering our entire introductory DC circuits course. By means of the menu program, a student could choose to work an unlimited number of problems -- and see complete step-by-step solutions to those problems -- at a time and pace of his own choosing.

The approach we took for the design of each module was to simulate as closely as possible a one-on-one student-and-teacher problem solving session. The "teacher" generates a succession of problems for the student to solve, offers "help" if asked, and at the end, provides the solution if needed. A student may work the problems on his own initiative, or be assigned a specific number of problems to work -- either as a standard homework assignment or as a remedial exercise because of difficulty in the course.

Our first modular program was on scientific notation and proved to be extremely successful. Freshmen students initially were somewhat reluctant to try the computer for the first time. But gradually, through faculty prodding and the efforts of a hard-working student assistant, students began to use our program on scientific notation and increase their mathematical proficiency.

Reinforced by this success, we increased our efforts and developed additional programs. For our introductory course in DC circuits, we have developed the following programs:

1. Scientific Notation
2. Prefixes
3. Ohm's Law
4. Voltage Divider
5. Current Divider
6. Equivalent Resistance
7. Thevenin's Theorem
8. Mesh Analysis

The program PREFIX is designed to provide student practice with common electrical prefixes such as: milli, micro, nano, pico, kilo, and mega. Questions and circuit parameters used in the questions are randomly generated. If the student answers incorrectly, the computer will offer hints and help until either the student answers correctly or -- after a predetermined number of misses -- the computer finally gives the correct answer to the student. A tally is kept of the student's score and number of attempts and can be stored in a file if desired. The decision whether to store a grade or not is left to the student. In this way, the student may work the exercises any number of times, developing his proficiency, and elect to store his results when his score is high. Once the data is stored, however, it cannot be changed or deleted by the student.

OHM's LAW is a program which provides student practice in solving single-source DC series circuits. The circuits and component values are randomly generated and are drawn on the screen using high-resolution graphics. Figure 2 shows a typical Ohm's Law problem being solved.

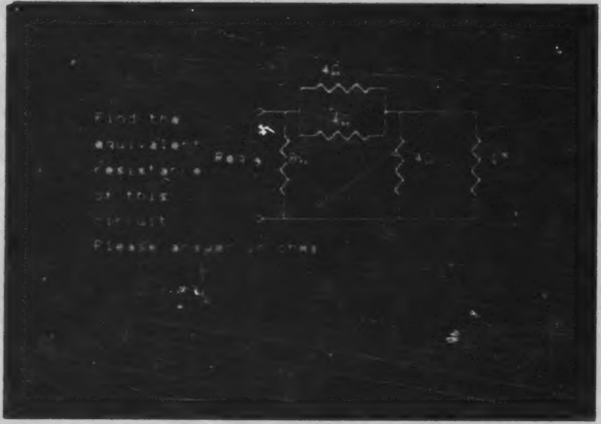
Figure 2. A simple Ohm's Law Problem for DC circuits. The computer will accept an algebraic operation as an answer.

The student is asked to solve for either the current, the voltage, or the resistance necessary to satisfy Ohm's Law. The circuit complexity varies from one to three series resistors after initially generating three single resistor circuits for the student to solve. If the student answers incorrectly, he is given the correct answer and the problem is worked correctly by the computer. The total number of student exercises, the number of correct responses, and the percentage answered correctly may be saved in a file as described earlier.

The VOLTAGE DIVIDER program provides practice for a student to use the voltage divider rule to solve for the voltage drop across one resistor in a series string of resistors. The CURRENT DIVIDER program is very similar -- using the current divider rule to solve for the current through one of the parallel branches. Simplifying series-parallel circuits and reducing them to their single equivalent resistance is covered by a program called EQUIVALENT RESISTANCE. Figure 3 shows a student solving a typical problem. In this program, circuit complexity varies from a two-resistor circuit up to a six-resistor series-parallel combination. The program begins with the simple two-resistor circuit, then progressively more complex circuits are generated as the student demonstrates proficiency. A close-up of the CRT display is shown in Figure 4.

The THEVENIN'S THEOREM program demonstrates the steps required to find the Thevenin equivalent voltage and resistance. The circuits may use either a voltage source or a current source,

Figure 3. A student calculates the equivalent resistance of a randomly generated electrical network. Circuit complexity increases as the student develops increased proficiency.



Find the
equivalent
resistance
of this
circuit
Please answer in ohms

Figure 4. A simple equivalent resistance problem. High-resolution graphics capabilities of the APPLE II greatly simplify the production of electrical schematic diagrams.

determined randomly by the computer. As with all programs, feedback is immediate and student results may be saved.

The most complex program in our DC circuits series is MESH ANALYSIS. This program will generate one-, two-, or three-loop DC electrical networks for the student to solve or to follow through in a tutorial manner. The program demonstrates how to assign loop currents, find the polarity of the voltage drops, write the mesh equations, and finally how to set up and solve the determinants for the values of the loop currents. As in all programs, the circuits and component values are randomly generated once the student specifies the number of loops desired in his circuit. If the student works a problem incorrectly, the computer displays the correct solution in a step-by-step demonstration. A unique option in this program allows the student to specify a particular circuit configuration and all component values. In this way, a student may completely design the circuit which is to be analyzed so as to duplicate a textbook problem or re-work classroom examples on the computer until they are thoroughly understood.

Our AC circuit series includes the following programs: (1) SINE, (2) TRANSFORMERS, and (3) POWER. The SINE program allows a student to calculate effective and peak values of sinusoidal waves. Practice is also given in determining the phase-angle between two sinewaves. Transformer turns ratio and transformer efficiency are the topics covered by the program TRANSFORMERS. The program on POWER provides student drill and practice in solving for power in AC circuits. Circuits of one-, two-, or three-components in series or parallel are generated randomly by the computer. Students are asked to solve for the power or power factor of these circuits. Complete solutions are provided to those students who answer incorrectly, and student results may be saved as described earlier.

Our introductory electronics series at present consists of three programs: (1) RECTIFIER, (2) TRANSISTOR BIASING, and (3) TRANSISTOR AMPLIFIERS. The program RECTIFIER provides students with practice in solving basic half-wave, full-wave, and bridge-rectifier circuits. Since these programs utilize transformers as input devices, students are required to understand transformer operation. Fixed-biased, voltage-divider biasing, emitter-biasing, and collector-feedback biasing are covered in a program called TRANSISTOR BIASING. In this program, the student may elect to solve a single type of biasing problem or may let the computer randomly generate the problems. The program TRANSISTOR AMPLIFIERS allows the student to solve for voltage gain, input impedance, and maximum undistorted output of common-emitter amplifiers or emitter followers. With the exception of the PREFIX program, all of the programs have the unique feature of accepting mathematical operations as their input. This means that the student no longer must have his calculator to calculate the answer to input into the computer. For example, a student may enter 20 x 60/(20+10) as his answer and the computer will calculate and display the answer of 40. The hierarchy of mathematical operations is the same as the standard Basic language calculations.

CONCLUSIONS

In conclusion, we feel that we are finally utilizing the unique capabilities available from computers to enhance student learning and understanding of engineering concepts. Students actually enjoy working with the computer as they learn. The concept of using the computer to enhance and expand the horizons of technical education is an idea whose time has surely come. The on-going advances in microprocessor-based computers and computer-generated graphics provide us with an exciting and challenging variety of new instructional techniques and strategies. The hardware to implement these strategies is readily available at low cost from a number of sources. It is up to each of us individually to use these new educational tools to their fullest potential.

A CASE STUDY OF AN EFFECTIVE EXTERNAL
EVALUATION MECHANISM

Arthur C. Moeller and Russell J. Niederjohn
Department of Electrical Engineering and Computer Science
Marquette University
Milwaukee, Wisconsin 53233
(414) 224-6788

INTRODUCTION

The Department of Electrical Engineering and Computer Science at Marquette University has recently completed a three-year project whose purpose was to achieve a significant improvement in the quality of its undergraduate educational programs by a substantial expansion of the small computer content in all facets of the curriculum. This project was supported by a grant from the National Science Foundation under its Comprehensive Assistance to Undergraduate Science Education program. The objectives of the Marquette CAUSE project were realized by the implementation of a tightly constructed plan designed to promote the inclusion of small computer principles and applications throughout the undergraduate curriculum. This plan consisted of four major components: (1) a faculty development program to increase their small computer background; (2) an undergraduate core and option course revision program to reflect more adequately the increasing pervasiveness of small computers; (3) a modernization and expansion program for the undergraduate small computers laboratory to permit the growth of meaningful hands-on experience; and (4) use of an evaluation mechanism to assess the effectiveness and quality of the activities proposed for the first three components and to redirect these activities constructively whenever appropriate. This evaluation mechanism included a vital external review component.

The evaluation program was planned to monitor and assess the various activities proposed under the Marquette CAUSE project and to redirect them as appropriate. It is anticipated that the evaluation process will be expanded in the next two years to encompass all undergraduate programs in the department. This paper will examine the evaluation program with particular attention to the critical external review component and will discuss its total impact on the department including its effectiveness in promoting the objectives of the local CAUSE project. The evaluation program as proposed received very favorable comments from the panels who reviewed the Marquette proposal. After just over one year of experience, outside consultants retained by the National Science Foundation to review the CAUSE program on a nationwide basis found the evaluation activity of the Marquette project to be strong and effective. Subsequent experience has served to strengthen these early expectations.

EVALUATION PROGRAM

Overall responsibility for evaluation activities under the Marquette CAUSE project was assigned to a senior member of the department faculty who was designated as the evaluation coordinator. He served as chairman of two groups appointed by the department chairman specifically for the purpose of assessing, redirecting, and promoting the project activities. One group, the CAUSE Evaluation Board, was recruited nationally and consisted of five members including top executives with computer expertise from large companies in Seattle and Chicago, the president of a small computer consulting firm in Milwaukee, and the chairman of a large academic department of computer information science in Cleveland. In retrospect, the caliber and composition of this group were critical in terms of its major impact on the project and the department. The other group, the CAUSE Faculty Committee, consisted of five senior faculty of the department who were chosen on the basis of their substantial involvement in the undergraduate core curriculum and their ability to realize curriculum modernization. The close interaction of these two groups and the insights which each derived from the other were instrumental in achieving an effective evaluation program. These groups also interacted very effectively with the project director. Both groups and the project director were very much impressed by the quality of the interaction.

The evaluation process was formally launched with a two-day site visit by the Evaluation Board in late 1978 shortly after the start of the project. The purpose of this visit was to evaluate the status of small computers in the undergraduate program prior to the CAUSE project and to review baseline data. This visit was followed by site visits at the end of the first and second years of the project and again at the end of the project. The typical site visit included a half-day of informal interaction with students and faculty usually in a laboratory setting; a half-day of formal presentation of project results and plans by the project director, the evaluation coordinator, and other faculty participating in particular phases of the project; and a half-day of give-and-take discussion among the members of the Evaluation Board, the members of the Faculty Committee, and the project director. This process led to a consensus assessment of progress in the various project activities as well as to specific recommendations for future directions of these activities. The Faculty Committee and the project director met with the Evaluation Board at all formal sessions except executive sessions at which evaluations and recommendations were prepared. They also had lunch and dinner together.

A final site visit will be made two years after the end of the project to assess the staying power of the computer modifications in the undergraduate courses and laboratories as well as to evaluate continuing enrichment of these curricular aspects. It is expected that the five-year experience of the Evaluation Board will culminate in a proposed evaluation process to be applied on a comprehensive department-wide and continuing basis.

Certainly in part because each site visit included extensive interaction with the Evaluation Board, the members of the Faculty Committee were very supportive of the project director in the following year and worked closely with him to implement the recommendations in the formal report of the Board. The NSF consultants noted that the Board was particularly important in raising questions of means-ends relationships and in looking for gaps and blind spots. They observed that the thoughtful responses of the members of the Board after each site visit indicated that they were indeed fulfilling this role. The consultants also noted that the evaluation techniques were exemplary in their integration with the decision-making structure of the project and that the Board aggressively examined the project at various stages in terms of the appropriateness of its goals and in terms of the validity of the approaches to achieve these goals. Evidence of the effectiveness of the Board includes specific changes made in the project in response to Board suggestions.

Areas examined by both the Faculty Committee and the Evaluation Board included the minicomputer and microcomputer content of core curriculum and computer elective courses, computer facilities and equipment, faculty programs to develop background in computer principles and applications, and primarily the impact on undergraduate students of computer course content and laboratory experience. All site visits were supported by detailed background materials prepared and distributed in advance by the project director and the evaluation coordinator with the assistance of the Faculty Committee. These materials included development activities, undergraduate course modifications and additions, undergraduate laboratory modernization and expansion, and undergraduate student enrollment data. Also included were copies of questionaires which all faculty of the department had completed and a copy of another questionaire to which a very high percentage of the senior students had responded. A detailed analysis of the responses was provided together with a comparison to baseline data obtained just before the start of the CAUSE project. These questionaires were judged by the NSF consultants to be of a higher standard than normally encountered in undergraduate curriculum evaluation. In a carefully integrated series of questions, both faculty and students were asked to rate their backgrounds in computer principles and applications, to anticipate how important computer background might be in their future work, and to indicate how strong their computer backgrounds might now appropriately be in light of the previous responses. Faculty were also asked to indicate how much use of computer principles and applications they were currently making in their teaching and research and specifically to note how much content of each course they were presently teaching involved computer material. Students were asked to indicate which university courses had strengthened their computer backgrounds and to what extent. Both faculty and students were encouraged to suggest specific means which they would find most effective in improving their computer backgrounds to the desired level.

Formal reports were prepared shortly after each site visit. These reports were submitted to the department chairman who forwarded them to the dean of engineering and to the academic vice president. These reports included copies of most materials prepared for the visit, minutes in summary form of the formal sessions, and most importantly specific assessments by the Evaluation Board of progress in each of the project components and recommendations with respect to future activities in each of these. Typical evaluations and recommendations from the April 1980 report are given below.

> "Faculty and course development has been substantial and impressive."
> "Modest improvement in laboratory

facilities has been realized, especially in the computer graphics area."

"Efforts toward enhanced faculty awareness of small computers should be continued throughout the University. These efforts will pay big dividends in improved programs. Such efforts sould certainly include Business Administration faculty."

"A five-year plan for continuing development of the computer sciences area should be drawn based upon a comprehensive statement of purpose and objectives. CAUSE has laid an excellent foundation in small computer technology. This plan should define relationships with other departments as well as project course developments, staff requirements, facilities, and resources. Such a plan should be viewed as a strategic plan which would be updated annually. Plans to obtain increased funding consistent with the five-year plan must be developed and funding efforts should be pursued internally and externally"

Also included in the formal report was a copy of the individual report written by each member of the Evaluation Board which contained his own reactions to the meeting, his personal assessment of progress, and his recommendations for future efforts.

IMPACT

The evaluation program described in this paper was very effective in terms of the Marquette CAUSE project, but it had an even more profound impact on the future of the total department. The external evaluation component was very critical to the success of the evaluation process. The members of the Faculty Committee were strongly influenced by the insights and experience of the Evalaution Board and were highly motivated to provide enthusiastic support for the many activities under the CAUSE project. The members of the Evaluation Board in turn were impressed by the quality and extent of the advance preparation for their meetings and expressed their gratitude for the consequent opportunity to make real contributions to the progress of the project. They were convinced that their contributions were important in shaping the course of the project.

Interaction with the Evaluation Board had significant impact on the development of individual faculty and students. For example, one senior faculty member spent his sabbatical year at the company directed by a member of the Evaluation Board working on a computer graphics design project in which both had become interested. However, from the point of view of the department, the impact of the Evaluation Board was enormous. All objectives of the CAUSE project were achieved, some to an extent and in directions not originally anticipated. The Evaluation Board acted as a catalyst in bringing together a number of faculty to form a computer science and engineering group. With the encouragement and guidance of the Board, these faculty developed a formal computer science and engineering option in the undergraduate curriculum of the department. With the support of the Board, this option was approved by university administration and resulted in a change of name to the Department of Electrical Engineering and Computer Science. The impact on undergraduate students continues to be large as increasing numbers elect the computer science and engineering option.

Topics

THE EFFECT OF TECHNOLOGY IN THE TRADITIONAL ARTS: COMPARISON AND CONTRAST OF FEMALE AND MALE MUSIC STUDENTS' ATTITUDES TOWARD EAR-TRAINING COMPUTER-ASSISTED INSTRUCTION (CAI)

E. Margo Dillard
and
Rosemary N. Killam

School of Music
North Texas State University
Denton, Texas 76203
817-788-2791

INTRODUCTION

Since 1977, ear-training computer-assisted instruction (CAI) has been used as an adjunct means of study for the students enrolled in the first two years of music theory at North Texas State University (NTSU). Several studies of students' attitudes have been conducted, but no investigation has previously been made of possible attitudinal differences between females and males. This paper presents the results of research into female-male attitudes toward CAI. At the end of each semester, an attitudinal questionnaire has been given to the students who use the lab. This questionnaire was developed at Stanford University, and later used and refined at the University of Delaware (see Appendix A).

A preliminary study done in October 1981, on questionnaires collected at the end of the previous spring semester, showed that of the population responding, 37% were females and 63% were males. The smaller percentage of females led to a preliminary hypothesis that fewer women used the CAI Lab; reasons for this small percentage, as well as the profile of both female and male students who chose not to use the lab, prompted additional investigation.

In consultation with Ms. Nancy Kunsak, a member of the NTSU Psychology Clinic staff, Ms. Dillard developed a separate questionnaire of 15 items (see Appendix B), structured to determine attitudes of students who made little or no use of the Lab. Questions were structured around three possible reasons for limited use: 1) fear of technology, 2) time availability, and 3) self-concept, with a final question asking for additional student comments. The previously- and newly-developed questionnaires (presented in Appendix A and B) were distributed at the end of the fall semester, 1981.

BACKGROUND AND LITERATURE

Data obtained from the questionnaires have significance for the general population of college students. Popular opinion holds that women are more apt than men to exhibit fear of technology. The attitudes of women in a traditional art toward CAI may offer an additional perspective on women's supposed fear of technology in higher education. Additionally, if musicians of either sex display fear of technology, computer use will need to incorporate strategies to lower such anxiety.

Those pursuing serious study of music at the college level have previously had little exposure to computer technology, although they generally have made a long-term, pre-college commitment to musical activities. In the United States, major conservatories and schools of music admitted women long before Ivy League universities and technical institutes did so. Thus, women have been able to receive the finest post-secondary instruction available in music, which has traditionally attracted many women to the field. The women engaged in career-oriented musical study have heretofore had as little reason as men in the field to be heavily involved in computer technology.

Currently, musicians are involved in an ever-increasing use of computers. The areas of music which exhibit the greatest incorporation of computer technology are composition, analysis, and CAI. Several studies have investigated student attitudes toward the use of CAI in music (Herrold, 1973; Killam and Lorton, 1974;

Hofstetter, 1979; Baczewski, 1980; however, these studies have not compared differences in attitude between females and males.

A limited amount of research on sex differences in attitudes toward CAI in other disciplines has been reported in the literature. Casner (1978) found a difference in eighth-grade female and male attitudes toward CAI in math. Beck (1979) found no significant difference in attitudes of high school females and males toward anticipated CAI in three different subject areas, although he found a significant difference by subject area content.

DESCRIPTION OF FACILITIES AND METHOD OF INVESTIGATION

September 1981, marked the fifth year of music CAI at NTSU, the second largest school of music in the nation. The current Music CAI Lab houses eight time-sharing terminals dedicated to student use. Students enrolled in 27 sections of undergraduate music theory continued to use the approximately 200 lessons available as an adjunct to their ear-training studies.

At the close of fall semester 1981, instructors of undergraduate ear-training were given both questionnaires (see Appendices A and B) and were asked to have each of their students complete and return the appropriate questionnaire. Two hundred and forty-five questionnaires, with sex of respondant indicated, were returned out of a population of 411, a 60% return rate. Of the 245 returned, 184 returned Questionnaire A and 61 returned Questionire B.

Nine copies of Questionnaire A and two copies of Questionnaire B were returned with insufficient identifying information to determine the sex of the respondant. The results of these questionnaires were not included in the statistical compilation, but the comments made on them were included in the summary of comments presented in Appendix C and D, respectively.

As shown in Table I, the percentage of females and males in the subject group which completed Questionnaire A did not vary significantly from the total population.

Table I

Percentages and Numbers of Students
In Each Population Group by Sex

Population Group	Females	Males	Total
Questionnaire A	34% (63)	66% (121)	100% (184)
Questionnaire B	25% (15)	75% (46)	100% (61)
Total Enrollment	32% (133)	68% (278)	100% (411)

The Questionnaires consisted of statements (see Appendices A and B) which students rated on a Likert scale from 1 to 7, 1 being "Strongly Agree" and 7 being "Strongly Disagree." Questionnaire data were encoded and analyzed through SPSS (Nie, et al., 1975). The F statistic was used to determine the significance in variance between female and male responses. If no significant difference in variance was found, female-male mean scores were examined, using a Two-tailed T-test, with pooled variance estimate. Answers to only two questions showed significant differences in variance; for those instances, the T-test using separate variance was employed. The .05 level of confidence was selected to determine significance.

RESULTS

The statement "I used the CAI Lab to practice ear training:" was placed at the beginning of both questionnaires (see Appendices A and B). For statistical purposes the responses "Never" through "Regularly" were assigned the numbers one through six, respectively. Data from both questionnaires were pooled, and frequency of use was examined. No significant difference was found between female and male responses. Out of a total population of 245, 233 responded to the question , with a resulting mean of 2.5 (2 = Once, 3 = A few times).

As shown in Table I, 184 students answered Questionnaire A. The results of this questionnaire are presented in Table II. Significant female-male response difference were found on statements 1, 3, and 13, as indicated by an asterisk.

Students took advantage of the opportunity to make comments on Questionnaire A. Their comments were sorted into five main categories by general subject matter: time availability, sound quality, curriculum areas, technical aspects, and miscellaneous. The comments were transcribed exactly as the students wrote them, and are presented in Appendix C. Sixty-nine students (37% of the group responding to Questionnaire A) chose to include comments, 25 females and 44 males.

As shown in Table I, 69 students answered Questionnaire B. Table III summarizes response means for Questionnaire B for both Fall 1981 and Spring 1981 surveys for comparison. As stated previously, the Spring 1981 survey data prompted this investigation.

Response on only one statement, number 3, showed a significant difference in female-male response. However, females displayed a different dispersion than males in their ratings of several statements. Statement 3, "I like working at my own pace at the terminal," showed a significant difference in means, 1.13 for females and 1.56 for males

Table II

Questionnaire A Response Statistics

Statement	Sex	No. of Responses	Mean	St. Dev.	F Value	2-Tail Prob.	Pooled Variance Estimate		
							F Value	Degrees of Freedom	2-Tail Prob.
*1. I have a good ear already.	Females Males	62 120	3.66 3.05	1.81 1.56	1.36	0.16	-2.37	180	0.02
2. I have too much work to do in my other classes.	Females Males	62 119	2.52 2.82	1.54 1.90	1.52	0.07	1.10	179	0.27
*3. My job consumes my spare time	Females Males	60 117	3.97 4.85	2.60 2.47	1.11	0.64	2.20	175	0.03
4. I do not have time to spend outside of class practicing ear-training.	Females Males	62 118	4.03 4.24	1.78 1.91	1.14	0.57	0.70	178	0.48
5. Computers are too complicated.	Females Males	62 118	5.37 5.89	1.89 1.75	1.17	0.48	1.84	178	0.07
6. I prefer practicing with a classmate.	Females Males	61 119	4.08 4.19	1.96 2.07	1.07	0.79	0.35	178	0.72
7. I could put my free time to better use in other activities.	Females Males	61 118	4.38 4.07	1.84 1.85	1.01	0.98	-1.06	177	0.29
8. I could never learn which buttons to push to make the thing work.	Females Males	62 118	5.84 6.16	1.76 1.57	1.25	0.30	1.26	178	0.21
9. The lab hours conflict with my class/work schedule.	Females Males	62 118	3.35 3.82	2.16 2.13	1.03	0.89	1.39	178	0.17
10. I was never very good in math.	Females Males	62 118	5.26 5.42	2.22 1.94	1.31	0.22	0.49	178	0.62
11. I don't like being taught by a machine.	Females Males	62 117	5.45 5.21	1.78 2.06	1.34	0.21	-0.77	177	0.44
12. Computers know too much about everybody already.	Females Males	61 118	5.85 5.83	1.59 1.81	1.30	0.26	-0.08	177	0.94
*13. I don't think a computer could make me hear better.	Females Males	62 118	5.63 5.41	1.39 1.89	1.83	0.01	(Separate Variance) -0.90	158.45	0.37
14. I am frustrated by highly technical things like computers.	Females Males	62 118	5.69 6.08	1.70 1.64	1.07	0.76	1.50	178	0.14
15. I don't like having my scores reported to my teacher.	Females Males	60 116	5.55 5.38	1.63 1.90	1.36	0.19	-0.50	174	0.55

* Significant Female-Male Response Difference

Table III

Summary of Means for Questionnaire B

Response Scale: 1 = Strongly Agree, 2 = Moderately Agree
3 = Slightly Agree, 4 = Uncertain
5 = Slightly Disagree, 6 = Moderately
Disagree, 7 = Strongly Disagree

		Fall 1981	Spring 1981
1.	I worked as hard answering questions in the computer lessons as I do in the classroom.	2.07	2.45
2.	I learned from the computer lessons as well as I would have learned the same lesson in the classroom.	2.92	3.37
3.	I like working at my own pace at the terminal.	Female = 1.13 Male = 1.56	1.86
4.	I would prefer competing with my fellow students in the classroom rather than working at computer lessons.	4.70	4.62
5.	Working with computer lessons is like having my own tutor.	2.70	3.16
6.	Thirty-minute sessions allow sufficient time to make adequate use of the material.	4.77	4.28
7.	I found the computer lessons too easy.	5.55	5.18
8.	I think working with computer lessons is an exciting way to learn.	2.93	3.20
9.	I found working at the terminal more frustrating than worthwhile.	5.30	4.65
10.	I would like to participate in another CAI course.	2.08	2.48
11.	I found the computer lessons too hard.	4.80	4.57
12.	The CAI system provides the student with more feedback than classroom instruction.	3.85	3.83

(1 = Strongly agree, 2 = Moderately Agree), and also in standard deviations, .35 for females and 1.12 for males, with a total of 60 responding.

On Questionnaire B, comments were included by 35 students (57% of the total), 10 females and 35 males. These comments were sorted into four main categories: time availability, sound quality, curriculum areas, and miscellaneous. The comments are presented in Appendix D.

DISCUSSION

Questionnaire A was designed to examine students' attitudes in three areas: fear of technology, time availability, and self-concept. The statements which dealt with fear of technology to some extent were numbers 5, 8, 10, 11, 12, 13 and 14. The responses to all these statements were similar, showing no significant difference between female and male attitudes. The overall population disagreed with all of these questions, showing a lack of intimidation or fear of any kind concerning computer use. Statement 8, "I could never learn which buttons to push to make the thing work," elicited the most extreme response, in terms of the scale "Strongly Agree" through "Strongly Disagree," of all statements on the questionnaire, with an overall mean of 6.06. Furthermore, the level of strength of disagreement on all seven of these statements was stronger than on any other statement, and was stronger than the equivalent level of strength of agreement on the four statements which showed means falling into the "agree" areas of the scale.

Five statements dealt with lack of time as a factor in limited use of computers: numbers 2, 3, 4, 7, and 9. These questions, except 3, showed no significance between females and males and response fell into either "Agree" or "Uncertain." Number 3, "My job consumes my spare time," showed a significant difference between females and males with females agreeing and males disagreeing, implying a greater tendency for females to have outside employment in addition to their studies.

Self-concept was measured in statements 1, 6, and 15, and was partially evident in statements 10 and 13. A significant difference between females and males occured. Females agreed significantly less than males with the statement "I have a good ear already," number 1. Additionally, the variance was significantly less for females than males on statement 13, "I don't think a computer could make me hear better." Although agreement with this question might indicate a fear of computers, given the overall profile of lack of fear of technology, the answers probably indicate self-concept. The difference in response variance arose from the fact that no women strongly agreed with the statement, and only 4.8% showed any level of agreement with the statement, while 16.1% of the males agreed to some extent. Thus, a significantly greater number of males seemed satisfied that their level of auditory ability was quite sufficient without CAI usage.

The statements on Questionnaire B were divided into two basic categories: those which showed a positive attitude toward computers and those which showed a negative attitude. Statement numbers 1, 2, 3, 5, 8, 10, and 12 reflected a positive attitude, whereas statements 4, 6, 7, 9, and 11 were negative to some extent. The students responding to this questionnaire agreed with all statements which were positive and disagreed with all statements which reflected a negative attitude, showing a preference for computer use. This response was compared to the response to the questionnaire completed at the close of spring semester 1981. At that time, the questionnaires were distributed primarily in the lab rather than the classroom, thereby creating basically the same population group as the Fall 1981 survey, students who made "occasional" to "regular" use of the lab. However, in the former group, a lesser tendency to agree with favorable statements was found, with more response falling in the "uncertain" range. On the Fall 1981 survey only one statement showed a significant difference in response between females and males. Statement 3, "I like working at my own pace at the terminal," showed greater agreement among females, reflecting a higher concern for use of time than males. The histograms in Figure I illustrate the remarkable homogeneity of female, as opposed to male, ratings.

Figure I

```
            No. 3

HISTOGRAM FOR FEMALES

CODE
        |
     1. |*********************** (13)
        |    STRONGLY AGREE
        |
     2. |**** (2)
        |    MODERATELY AGREE
        |
        |******|******|******|******|******|
        0      4      8      12     16     20
        FREQUENCY


            No. 3

HISTOGRAM FOR MALES

CODE
        |
     1. |********************* (30)
        |    STRONGLY AGREE
        |
     2. |******** (11)
        |    MODERATELY AGREE
        |
     3. |* (1)
        |    SLIGHTLY AGREE
        |
     4. |** (2)
        |    UNCERTAIN
        |
     7. |* (1)
        |    STRONGLY DISAGREE
        |
        |******|******|******|******|******|
        0      10     20     30     40     50
        FREQUENCY
```

Over one-third of the students responding to both questionnaires chose to make additional comments, which were divided into five categories (see Appendices C and D). The largest category was time availability for CAI use, with 31% of the comments involving this topic. As can be seen in the comments, a great number of students requested expanded hours of computer availability. (The lab is presently open only 40 hours a week, due to lack of funds for lab monitors.) Forty-seven percent of these time comments were made by females, as opposed to 53% by males. Comparison of these percentages to the overall population distribution of 32% females and 68% males supported the idea of greater concern of females for use of their free time.

SUMMARY AND CONCLUSIONS

The study undertook an investigation of attitudes of female and male music students toward use of computers. This population has no traditional association with technology, thus would not be a self-selected group with favorable attitudes toward computers. A special questionnaire was devised and administered to check attitudes among those who made little use of the CAI system, on the premise that this subgroup would be more likely to display negative attitudes toward computers. Neither females nor males exhibited any fear of technology. The major area comments from both questionnaires concerned lack of available lab hours. Furthermore, women indicated greater constraints on their out-of-class time (caused by part-time work) than men, and women showed less confidence in their ear-training skills than men.

A new questionnaire is being developed to measure, in more detail, students' time commitments to outside activities and their self-concepts. This questionnaire will be administered at the close of Spring semester 1982, and results will be reported at the conference.

Both women and men displayed positive attitudes toward computer technology. The lack of fear of computers among a large undergraduate population in a discipline with no historic involvement with technology is most encouraging and bodes well for the expansion of computer-assisted instruction to new disciplines and student groups.

ACKNOWLEDGEMENTS

The authors express their appreciation to Ms. Nancy Kunsak, of the NTSU Psychology Clinic staff, for consultation concerning psychometrics.

REFERENCES

Baczewski, P. Experience and Evaluation: Ear Training CAI in Action. Proceedings, Texas Music Educators' Association Conference, San Antonio, Texas, 1980.

Beck, J.J. The Effects of Attitude on Anticipated Computer-Assisted Instruction in Selected High School Courses of Study. Association for Educational Data Systems Journal, Vol. 12, No. 3, Spring 1979.

Casner, J.L. A study of attitude toward mathematics of eighth grade students receiving CAI and students receiving conventional classroom instruction. (Doctoral dissertation, University of Kansas, 1977). Dissertation Abstracts International, 1978, 38, 7106A.

Herrold, R. Computer-assisted Instruction: A study of student performance in a CAI eartraining program. Unpublished DMA Final Project, Stanford University, 1973.

Hofstetter, F. Controlled Evaluation of a Competency-Based Approach to Teaching Aural Interval Identification. Proceedings, Association for the Development of Computer-Based Instructional Systems, San Diego, 1979.

Killam, R.N. and Lorton, P.V., Jr. Computer-assisted Instruction in Music: Ear-training Drill and Practice. Proceedings, Fifth Conference on Computers in the Undergraduate Curricula, 1974, pp. 263-273.

Nie, N.H., Hull, C.H., Jenkins, J.G., Steinbrenner, K., and Bent, D.H. SPSS: Statistical Package for the Social Sciences. (2nd ed.) New York: McGraw-Hill, 1975.

Appendix A

Ear Training Class____________ Soc.Sec.Nr. (Needed) ___________

Date__________________ Sex (Needed) __________

Name (Optional) _______________

STUDENT EVALUATION FORM
NTSU COMPUTER-ASSISTED INSTRUCTION (CAI)

Please circle the most appropriate response:

I used the CAI Lab to practice ear training:

Never Once A few times Occasionally Frequently Regularly

If you answered "Never", "Once", or "A Few Times",

Please complete Questionnaire A

If you answered "Occasionlly", "Frequently", or "Regularly",

Please complete Questionnaire B

QUESTIONNAIRE A

Please read each statement and circle the number of the scale that best describes your feelings.

SCALE: 1 Strongly Agree
2 Moderately Agree
3 Slightly Agree
4 Uncertain
5 Slightly Disagree
6 Moderately Disagree
7 Strongly Disagree

1. I have a good ear already	1	2	3	4	5	6	7
2. I have too much work to do in my other classes.	1	2	3	4	5	6	7
3. My job consumes my spare time.	1	2	3	4	5	6	7
4. I do not have time to spend outside of class practicing ear-training.	1	2	3	4	5	6	7
5. Computers are too complicated.	1	2	3	4	5	6	7
6. I prefer practicing with a classmate.	1	2	3	4	5	6	7
7. I could put my free time to better use in other activities.	1	2	3	4	5	6	7
8. I could never learn which buttons to push to make the thing work.	1	2	3	4	5	6	7
9. The lab hours conflict with my class/work schedule.	1	2	3	4	5	6	7
10. I was never very good in math.	1	2	3	4	5	6	7
11. I don't like being taught by a machine.	1	2	3	4	5	6	7
12. Computers know too much about everybody already.	1	2	3	4	5	6	7
13. I don't think a computer could make me hear better.	1	2	3	4	5	6	7
14. I am frustrated by highly technical things like computers.	1	2	3	4	5	6	7
15. I don't like having the scores reported to my teacher.	1	2	3	4	5	6	7

16. Use the back of this sheet to make any additional comments you wish concerning the CAI program, especially any additional types of lessons you would like to see available.

Appendix B

Ear Training Class ____________ Soc.Sec.Nr. (Needed) ____________

Date ________________ Sex (Needed) ____________

Name (Optional) ____________

STUDENT EVALUATION FORM
NTSU COMPUTER-ASSISTED INSTRUCTION (CAI)

Please circle the most appropriate response:

I used the CAI Lab to practice ear training:

Never Once A few times Occasionally Frequently Regularly

If you answered "Never", "Once", or "A Few Times",

Please complete Questionnaire A

If you answered "Occasionlly", "Frequently", or "Regularly",

Please complete Questionnaire B

QUESTIONNAIRE B

Please read each statement and circle the number of the scale that best describes your feelings.

SCALE: 1 Strongly Agree
2 Moderately Agree
3 Slightly Agree
4 Uncertain
5 Slightly Disagree
6 Moderately Disagree
7 Strongly Disagree

1. I worked as hard answering questions in the computer lessons as I do in the classroom. 1 2 3 4 5 6 7
2. I learned from the computer lessons as well as I would have learned the same lesson in the classroom. 1 2 3 4 5 6 7
3. I like working at my own pace at the terminal. 1 2 3 4 5 6 7
4. I would prefer competing with my fellow students in the classroom rather than working at computer lessons. 1 2 3 4 5 6 7
5. Working with computer lessons is like having my own tutor. 1 2 3 4 5 6 7
6. Thirty-minute sessions allow sufficient time to make adequate use of the material. 1 2 3 4 5 6 7
7. I found the computer lessons too easy. 1 2 3 4 5 6 7
8. I think working with computer lessons is an exciting way to learn. 1 2 3 4 5 6 7
9. I found working at the terminal more frustrating than worthwhile. 1 2 3 4 5 6 7
10. I would like to participate in another CAI course. 1 2 3 4 5 6 7
11. I found the computer lessons too hard. 1 2 3 4 5 6 7
12. The CAI system provides the student with more feedback than classroom instruction. 1 2 3 4 5 6 7
13. Use the back of this sheet to make any comments you with concerning the CAI program, especially any additional types of lessons you would like to see available.

Appendix C

Sex of respondants is indicated in the left margin (M = male, F = female, NI = Not Indicated) Comments are transcribed exactly as written by the students; spelling and grammatical error are not corrected.

TIME AVAILABILITY

M I think it is a good program, I would prefer later hours.

F It takes much too much time to work on the computer. It is more efficient to listen to tapes.

F The hours opened are not enough. Stay open later than 6:00 and each person should be allowed 1 hr. on the machine instead of 30 minutes. Get the melodic & rhythmic dictation working too.

M Regardless of my answers on the front, I think the CAI lab is a super idea for those who need it or have time to fit it into their schedule.

F I would use the lab if the hours were different. I find I have class conflicts. I do enjoy the computer setup and it has helped me in the past. I intend to continue using it.

M I never had any available time to use the CAI Lab. That is the only reason.

M Because of many new responsibilities taken (job, etc.) I was hard pressed to schedule my time, so I decided for this semester that I just wouldn't use the CAI. I plan to next semester for I used it last year and it helped me fairly much.

F I used the system a lot last semester and know that it did help me. Due mostly to time limitations, I haven't been able to use CAI very much this term, and I regret my failure to spend time in CAI. I hope I'll be able to use the computer in the Spring and Summer to help me skills.

M It is hard to make the scheduled times in the lab. I'm sorry that I took more hours than I could handle so finding time (when I wasn't exhausted was hard. I have to spend as many free hours as possible making reeds, as I am an oboeist.

M A. I think it would be good if CAI was in an easier place to "drop in to". I would do it if it was on the way.
B. I dislike all grades going to the teacher. We should be able to choose if we want the teacher to receive. Perhaps a simple yes-no question at the end of the program could accomplish this.
C. Try to extend the number of hours & number of terminals. The 2 times I went to CAI, once it was closed & once it was full. I'm a businessman, and my time is valuable. I can't spend time hanging around waiting for open terminals, or for a monitor to come around.

M The lab hours I would like to use the CAI are usually after the lab closes. I personally would rather not use the system until later at night.

F I could find no specific time that I could practice sight singing in the lab. So, I used my piano practice room for practicing sight singing. Usually, weekends were the best time for me. I feel I got in plenty of practice on my own.

M I didn't use CAI because I didn't feel I needed it, not to mention lack of time.

F hours available need to be increased--

M I believe that the 30-minute slots do not allow enough time for the student to really accomplish anything. I believe the limit of time scheduling should be increased.

M I didn't get the opportunity to use the CAI. I like to work with computers & yet I had too many hours so I couldn't work with the CIA.

F It takes too long to go through one lesson.

F The computer is great -- when time is available. I wish I could buy my own to have at home.

F My teaching hours conflict and I can teach myself how I know I can learn the eartraining info.

F I would have used the lab, but since I commute it was not possible. This is mainly due to your poor choice of lab hours. If you were open in the morning, I could have used it. I needed to use the lab too. I feel I could have improved my ear if I could have.

F The CAI Lab has been a good, fun experience -- The few times I've been there -- The hours conflict with my schedule, however - and the computer sounds train me to hear that sound & I can't "hear" as well on a "real" instrument (such as piano) after listening to computer tones -- it has helped though & was fun!

M I practice on my own because the lab conflicts with most classes.

M The lab needs to be open more hours.

F I would like to see more nightly hours.
Also the biggest problem I had when working with the computers last was that they were very out of tune.

SOUND QUALITY

F I did not like using the main CAI system. It is very hard to hear the chords & other notes. I feel the apple is an excellent system and much better. I usually used the apple programs 3 times a week on a regular basis.

M I heard from a couple of people that most of the things the computers played (when playing chords or more than one note at a time) were out of tune and distorted.

M The sound quality of the tones are too difficult and the hours stink. The tapes in the library use real instruments and are more often available.

M I much prefer to hear chord progressions and other ear training exercises played on a piano in a room. The computer tones are not always clear. Headphones give me a headache.

F I had a hard time trying to hear things I was looking for.

F I'd like to see a better tone generator on this machine because the sounds it make for pitches are rather unpleasant

M It is very different (much harder) to hear an organ play chord & etc. than a piano. (For me at least it is)

M My one complaint is that the tone production of the computer is extremely erratic. Sometimes I really have trouble distinguishing pitches on the computer.

M The electronic tones really get on my nerves!

F It was always too difficult to figure out what the lesson I needed was called on the computer. Also I had a hard time hearing the chords when they are played on the computer. All the notes seemed to blend in with each other. It's easier for me to practice with a classmate.

M I think the CAI is a good thing, but there are still a few wrinkles which need to be fixed. The note-producing part of the computer is still not clear and precise, and sometimes their questions are comically below any humans range. For me, working with a real piano is better.

F When I went to the lab -- there were many pitches so out of tune that I got them wrong where as normally I'd have gotten them right.

M tone colour of computer sometimes difficult to hear
ex. minor-dimished

M Tuning of the machine is a consistant problem.

NI I don't know what kind of instrument the CAI lab uses for ear training, but is a very annoying sound and also the pitches are indistinguishable too often because the tones overlap each other too much. Since we, as students, have to take TPE and all of the finals on a piano, I strongly suggest that the recordings for the ear training would be done on a piano.

CURRICULUM AREAS

M The part which would help me most would be harmonic dictation, but I think the chord progression should be slowed down a bit.

F I don't believe that the lessons on a chord progression correspond with class lessons. They are in some cases far beyond what we are learning.

M I would like to see types of dictation available.

M The reasons I didn't use the CAI Lab this year were:
1) The procedure for melodic dictation ie actually telling the computer which note & rhythm you heard, was too slow. On the other hand, 2) the harmonic dictation tapes gave the chord so rapidly that I just couldn't keep up.
In general, I used the CAI Lab a great deal last year & enjoyed it very much. Also, I like the CAI Lab concept very much, & thorough support it.

M I wish there were more lessons on secondary dominants that didn't also cover advanced harmony (augmented chords, etc.). In other words, correlate the lessons with the book. The keyboard is frustrating to those of us who do not know how to type. Sometimes the range of the tones make the bass note inaudible (when all voices are low).

F I do believe the CAI Lab is an excellant opportunity to help with intervals and most basic chords. But it seems when it got into the melodic dictation I could never figure it out. And a lot of the sophomore level programs are messed up and that ends up confusing me more than helping. But I here that's being worked out. I have dislexia of the ear so nothing really helped.

M More melodic exercises

M a) Request: more harmonic dictation lessons, with helpful hints to show me how to to hear and think about progressions.
b) Perhaps a booklet describing mostly how to operate CAI & apple -- from the bottom up for those who know nothing.

TECHNICAL ASPECTS

M When I used the computers (3 time of hr. or more in one sitting) I found it hard to figure out what keys to push and if I got a question wrong usually I couldn't figure out why.

NI I think the keyboard buttons should be larger, and it would be a lot easier if they were color coded.

M (First of all, this evaluation is a joke.) From my own experience with the CAI lab, I found that it did not meet my needs. I had trouble learning to use the machine. When I finally mastered the art of programming the thing. I could not distinguish the varying sounds especially in the melodic & chordal parts of the program. Some pitches seemed to be others, pitches changed themselves during their sounding, and none of the chords seemed in tune. I also did not find the programs to be challenging. I just felt that for myself, the CAI did nothing to supplement or help my class work.

F I became very frustrated with the programming falacies of the system. Especially when I was given wrong information (ex. telling me I was wrong when the computer later told me the same answer). Also the length of delay on the melodic dictation excersises due to using the old Hubert Packert or whatever.

MISCELLANEOUS

M I think the hall leading up to the door is very attractive. Why didn't you ask a relevant question like that?

M I prefer to practice ear-training with myself.

M I'm sure the CAI progam is effective even though I'd rather not use it if I do need to hear intervals or cadences or harmonies, I'd rather play them on a piano or have someone play them for me. Just personal preference.

M Ear training can be obtained through using you instrument as a guidline for locating intervals and chord spellings.

M Expand the Apple program!

F I may have a slight mental block against the CAI lab, but I really don't feel intimidated by it. I'm sure the problem is just getting myself over there. I do realize that if I expect to get much further in eartraining, I'll have to spend some time there.

M In my opinion, everything in the world is becoming too dehumanized. Should we let music, the only personal expression everyone literate or illiterate can understand, become dehumanized, also. I think not!

M CAI Lab is good for those who need it but it should not be required. A computer can't teach you in as many ways as a human being can.

F I was too lazy to go by.

M I have had training with computers and I would enjoy programming music on to some of these computers. Is this possible or feasible?

M The main reason I did not use the CAI Lab is that I just did not feel that I needed it. If my grades in ear training suffered because I didn't use it it was only very slightly.

M I started out thinking that CAI was "optional", and procrastinated from going to sign up. When I found out that I was being required to use CAI, I found it was too much to catch up or I didn't bother. I did go once, and it was OK; my own choral experience has helped me greatly in eartraining, since I would sing alto, tenor, and bass. I can't say how much more it would have helped since I didn't go, but I know that my skills were on par with those who did go (except keyboard playing). After school I go to work, and since I signed up late, it was difficult to get a "good time" but thats more of a cop-out attitude. I got started on the wrong foot and stayed that way.

M I find that trascribing solos and tunes off records is a more productive & intense method of improving my ear at this point. The lessons at the C.A.I. don't seem too practical past the beginning level.

F The only reason I find I don't go to the lab more often is that there is no grade (I.E. daily homework) given for it. I'd like to think I had enough self-discipline to use the reward of a better ear as an incentive, but that's just not the case.

M It helped me immensely the few times I went.

F At the end of this semester I strongly regretted not going to the lab. It seemed to me that those who had gone continually were way ahead -- especially in the I-V-IV etc. I plan to be as regular as possible next semester -- It really makes a difference.

F I am interested in the program, would like to try, it haven't the opportunity to use.

M My ear seems to tire faster on C.A.I. than on piano.

M My reasons for not using the CAI lab have nothing to do with computers being difficult to operate. I think the CAI program can be extremely beneficial to the student who desires outside help in ear-training. The other think I can think of outside of the program is that while the computer corrects the student when he answers incorrectly it cannot always make clear to the student what his weaknesses are. (At least not as immediately as an instructor.

M The one time I went to the CAI Lab, I enjoyed the lessons very much. I was motivated to go only because my theory teacher deemed it a class requisite. The fact that I haven't gone back (yet) has no reflection on my opinion of the system. I just didn't go.

NI There needs to be alternatives to the computer. I have worked with reel to reel tape machines in the past using recorded piano examples of Harmonic Progressions. This to me is much more helpful and an easier way to learn. The program was very gradual and had a workbook to go along with it with all the answers to study. Most examples were from 7 to 10 chords long.

NI I used the apple frequently.

Appendix D

Sex of respondants is indicated in the left margin (M = male, F = female, NI - Not Indicated). Comments are transcribed exactly as written by the student; spelling and grammatical errors are not corrected.

TIME AVAILABILITY

M The CAI system is a great learning device. I only wish I took advantage of it more. Perhaps scheduled class time in the lab would be beneficial.

M I whish that the module in Bruce Hall could be tied in so that the print out would show the use of the facility. If one does not recieve credit for the terminal in Bruce Hall, what use is it for those who need credit but are unable to sign up for a terminal at a percise time. Musicians don't have sufficient time during the day to keep an exact schedule when it comes to practice.

M If I had more time available to use the computers I would because they help tremendously. PS. Thanks for the good selection of hours open.

F Melodic dictation
Would like longer lab hours. example: Open on Monday mornings

F Most of the times that I have time to go to the C.A.I. lab, the lab has been closed. If the lab were open more, I could have gone more

F Whenever the 1/2 hr. came - the lesson you'd been working on was erased. Also, it didn't record a lot of the lessons - the teacher didn't have proof of the lessons done. More melodic lessons - they never worked.

M I feel I really learned when I went to the lab. Unfortunately, however, I found it hard to find time in my schedule to use the lab alot. I hope to use if more next term however.

M Used the lab a great deal at the beginning of the semester but stopped because I couldn't spare the time.

SOUND QUALITY

NI The only problem seems to be intonation. Sometimes the computers.

M The quality of sound could be greately improved, especailly in the harmony lessons. It would be very helpful if lessons were not stopped in the middle of a lesson just because the 1/2 hr. has ended, unless others are waiting to use the terminal.

M I visited the CAI lab more during the beginning of the semester than during the end. One problem I had with the computer was its bad tuning.

F The chords are often out of tune.

M Many of the chords are out of tune! Sometimes chords are played in such a low range, they are difficult to destinguish. (they sound mushy)

F Sometimes range in bass too low to be distinguishable and helpful (e.g. bass identification)

M Need to straighten out pitch control on computer also need to get the recording of lessons straightened out

M Something needs to be done about the tone of CAI. Sometimes its so distorted that its hard to distinguish the pitches in a chord.

F Please make an effort to correct errors in programs. The tuning is sometimes dreadful & drives me out of the lab. I hope the computer is replaced without delay.

M Clean the "mud" out of the sound system, s'il vous plait?

CURRICULUM AREAS

M I wish you all would get your NHT & Melodic exercises working; you modulation exercises are to unballanced w/difficulty (i.e. Modulation w/V/V, the exercise has ii_7 &

other diatonic7 chords). Arnold is very good because; you see what you hear, and you hear everything much clearer.

F I would like to see more rhythmic dictation -- I realize you are workin on that. I feel that it has helped me improve this semester. Thanks.

F Some of the lessons seemed hard. It was fun at times, and at other times it upset me because I would miss a lot of questions. I think the computer system helps people who do not have someone to work with.

M More harmonization & modulation drills!!!!!

F I would like lessons discriminating Maj, min, dim, aug chords chords when played Block w/inversions - without Sop & Bass sensing. Those lessons which have bass, sop, & quality sensing are fine, but Lessons C11-C13 - don't help because the chords are arpeggiated & in root position.

M I started using the CIA Lab quite often at the baginning, but because of the lack of jazz lessons, I stopped

M I would like lessons which tell you exactly what to listen for. For Example, I would want to hear which voicing patterns are responsible for a a cadence's being perfect or imperfect.

MISCELLANEOUS

M Greg Ritchey did a good job working in the lab.

M Would like credit for Bruce Hall completed lessons - most of my time is on that terminal.

M I like the idea of a terminal available to the music students of Bruce Hall

M I wish I could do some of them.

F The computer never records the lessons completed, therefor, I stopped going to the lab.

M My most disappointing experiences with the lab occured on a regular basis when my teacher received his computer printout of the lessons I had completed. I cannot remember a single instance where the computer had accurately recorded the lessons I had completed.

M The CAI system is a great way to get things done that couldn't be done in class. It sometimes gets frustrating when there are bugs in the system, however.

M Your monitor Greg Ritchey did a good job in handeling the computers.

M CAI is good because I can go at my own pace doing an excercise that I need work on. Continue the program!!

M I would like the computer to give me a list of my lessons completed.

M I think CAI Lab is a great opportunity for improving your ear. I suggest they get more terminals because at certain hours they are all filled up. I also suggest that ear training classes be required to attend the CAI Lab once a month.

M I love it! I found my ear train increase quickly using it. I notice it in everyday use. If there would be some way of singing the lessons with out feeling like your disturbing other people that would be nice. If there was just a class for computer training, I would surely take it.

COMPUTER LITERACY: PRACTICAL NECESSITY

Colonel Thames F. Jackson, Jr
Administrator
Wilford Hall USAF Medical Center
Lackland Air Force Base, Texas

Major Timothy C. McKee
Chief, Systems Division
School of Health Care Sciences
Sheppard Air Force Base, Texas

Abstract

Prior to 1980, the introduction of computers into the Air Force health care administration process had been rather slow. Several new programs--The Uniform Chart of Accounts (UCA), Tri-Service Medical Information Systems (TRIMIS) Program, and Air Force Microcomputer Program--will drastically change this situation in the immediate future. In 1980 there were very few Air Force Medical Service Corps (MSC) officers who had any practical experience with computer systems. This paper describes the evolutionary development and implementation of a computer literacy block of instruction for the Basic Course in Health Service Administration in an effort to provide new MSC officers sufficient background in computer systems to cope with this imminent challenge.

1. INTRODUCTION

1.1 BACKGROUND:

For sometime now the day-to-day use of computers in the routine administrative operation of health care delivery activities has been something to be anticipated and planned for in the future. The anticipated future events are fast becoming current realities as the TRIMIS program to provide health care related computer systems comes to fruition, the UCA computer systems become a reality and the microcomputer computer initiative of the Air Force gets underway.

As this oncoming wave of computer systems drew closer, it became apparent that there were very few individuals assigned to our Air Force health care related activities who had any experience using computers. Discussions at our School of Health Care Sciences on the criticality of this fast closing need resulted in the development of an applied computer literacy block of instruction within the Basic Course In Health Services Administration.

1.2 SCHOOL OF HEALTH CARE SCIENCES:

The Air Force School of Health Care Sciences (SHCS) has the mission of instructing in the medical specialties and allied health care sciences to prepare competent professional and technical personnel for the Air Force Medical Service. Located at Sheppard Air Force Base, Texas, the School graduates more than 7,000 highly trained medical technicians each year. The school offers a Basic Course in Health Service Administration designed to prepare junior Medical Service Corps (MSC) officers for administrative positions in Air Force medical treatment facilities The eleven week course provides technical instruction in the areas of resource management, personnel administration, medical logistics, and patient affairs. Patient affairs is a functional description unique to the military which incorporates patient administrative activities including patient records management, ward clerk responsibilities, admission and disposition functions. release of medical information, and other related responsibilities. The majority of the students were ill-prepared to function effectively in a computer systems oriented environment. Although recent college graduates had some computer exposure, it was incomplete and inadequate to meet the demands of the job situation. It was in this situation that the decision was made to include 40 hours of applied computer literacy instruction in the Basic Course in Health Services Administration. There were two problems. The staff of the SHCS neither had the expertise to develop and teach the computer literacy block of instruction, nor possessed the necessary computer system hardware to support such a project. Although they had experience using PLATO to develop and present computer assisted instruction programs, they had no experience in teaching the use of operational support computer systems At that point in time the realization, among those involved with this project, became very acute that Dr. Molnar's statement on computer literacy being the next great crisis in American education (1) had become a reality for the Air Force Medical Service.

1.3 WILFORD HALL USAF MEDICAL CENTER:

The Wilford Hall USAF Medical Center (WHMC) is a one thousand bed teaching medical center which provides primary health care support to the Air Force Military Training Center at Lackland Air Force Base, Texas, and referral support for the 5th Department of Defense Region (Texas, Oklahoma, Arkansas and part of Louisiana). WHMC was fast entering the computer age, had a computer systems staff, and had experience using microprocessors.

1.4 THE CHALLENGE:

An agreement was reached between the Commander of the SHCS and the Commander of the WHMC for the WHMC staff to assist the School staff in developing and presenting the computer literacy block of instruction for the Basic Course in Health Services Administration. The SHCS formed the skeleton of a Systems Division to support the introduction of computer systems into their curriculum. The WHMC, with some assistance from personnel of the Air Force Medical Services Center, put together a "road show" of instructors, software, and hardware to provide interim capability until the School could provide the computer systems instruction from within its own resources.

Curriculum content was developed by Colonel Thames Jackson who had significant experience in systems design and systems operation along with developing training courses in support of operational computer systems. Details of programming exercises were developed by Staff Sergeant William Norrish, a computer systems technician. Captain Jerome Luby of the Systems Division, Air Force Medical Service Center (AFMSC), contributed the details on computer hardware and information on the coming computer storm via the TRIMIS, UCA, and Air Force Microcomputer Programs. Master Sergeant Michael Bollinger, also of the AFMSC, developed the Air Force ADP requirements development, and authorization procedure details. Major Timothy McKee provided support of the program at the SHCS. Much behind the scene work was done by the data automation staff at WHMC headed by Major Michael Lyons and Mr. Reynaldo Morales. Actual classroom instruction was performed by Jackson, Norrish, Luby, and Bollinger. All of this could not have been accomplished without the strong advocacy of Brigadier General Donald B. Wagner, Commander of the AFMSC and Chief of the AF Medical Service Corps; Brigadier General (now Major General) K. Vandenbos, Commander of the WHMC; and Colonel Richard Yeomans, Commander of the SHCS.

The SD Systems 200 microcomputer was selected to provide actual computer support.(2) The SD-200 was selected because it was "portable" and available at WHMC. This microcomputer had 256K bytes of internal memory, two double density 256K disk drives, a 600 character per minute daisy wheel printer, and would support five simultaneous users (primary CRT plus four additional CRTs, each using 48K of internal memory). This hardware along with spare circuit boards, cables, printer paper, a tool box, and miscellaneous support items were loaded into an Air Force twelve passenger van at WHMC and driven to Sheppard Air Force Base the day before each presentation of the computer literacy block of instruction. The hardware configuration was assembled, checked-out, and used to support the instructional effort. When the computer literacy instruction was completed, the configuration was disassembled, loaded back into the van, and returned to WHMC.

2. CONTENT

2.1 OVERVIEW:

The computer literacy course content was designed to provide an introduction to the use of data automation in an Air Force health care delivery environment and then develop an appreciation and understanding of its related technical and support aspects The general approach followed throughout the course was lecture/demonstration of a topic followed by extensive hands-on use of a terminal in a laboratory environment. The class was divided into teams of three or four students with the more experienced students being spread among the teams to function as peer teachers The theme of an admission, disposition and transfer (ADT) application to provide inpatient accountability and control was selected as a basis for exercise development. The students had already completed the patient affairs block of instruction (which includes the concept of inpatient accountability and control), so they were already familiar with the general ADT functions. This concept provided consistency and familiarity as the students developed each phase of the application, each new exercise building upon the previous exercise. Although not designed as a course in computer programming, programming using the Basic language was used to give a general understanding of how a computer user communicates with the computer via an application program. Most of the exercises were developed with parts being mandatory (necessary for the next exercise to build on) and optional (to provide an additional coordinated challenge to teams that completed the mandatory items before the laboratory time period expired). An outline of the computer literacy block of instruction as it finally evolved is as follows:

- Introduction to Medical ADP
- Systems Analysis
  - -- Exercise 1: Flow Charting
- Air Force ADP Requirement Development/Authorization Process
  - -- Exercise 2: The Mini Data Automation Requirement
- What is a Computer Program?
- Introduction to the Editor
  - -- Exercise 3: Using the Editor
- Basic I: Introduction
  - -- Exercise 4: Registering a Patient
- Basic II: Editing
  - -- Exercise 5: Editing
- Basic III: Control Loop
  - -- Exercise 6: Control Loop
- Basic IV: Top Down Design/Stubbing
  - -- Exercise 7: Stubbing an ADT System
- Basic V: Access Control
  - -- Exercise 8: Access Control
- Basic VI: Files
  - -- Exercise 9: File Development
- Documentation
- Project Management
- System Back-up and Recovery
- Data Privacy
- Contractor Relationships
- Hardware Maintenance
- Introduction to an ADT System
  - -- Exercise 10: Use of an ADT System
- Overcoming Resistance to Change/System User Training
- Data Base Management System (DBMS) Fundamentals
  - -- Exercise 11: Use of a DBMS
- Critique

In addition, two special enrichment evening sessions were held in an informal environment--usually at the Officers Open Mess--to facilitate discussion. The topics were:

- Air Force/Tri-Service Medical Information Systems (TRIMIS) Program. (The overall data automation program for the Air Force health care delivery arena.)

- "Who Wants To Be Moses?" (Current and near future data automation technology, the problems expected, and the leadership required in the environment it produces.)

Figure 1 is a schedule to show the arrangement of the subject areas in the instructional block.

SCHEDULE

MON	TUE	WED	THU	FRI
RESERVED FOR TESTING	WHAT IS A PROGRAM ? INTRO. TO EDITOR -EXERCISE #3 USE OF EDITOR	-EXERCISE #5 EDITTING	BASIC IV TOP DOWN DESIGN / STUBBING	PROJECT MANAGEMENT DATA PRIVACY CONTRACTOR RELATIONS HARDWARE MAINTENANCE BACKUP/RECOVERY
INTRO. TO MEDICAL ADP -CLASS OBJECTIVES -INTRO. TO COMPUTERS -COMPUTERS IN HEALTH CARE	BASIC I INTRODUCTION	BASIC III CONTROL LOOP(MENU)	-EXERCISE #7 STUBBING	INTRODUCTION TO ADT -EXERCISE #10 ADT
		LUNCH		
SYSTEMS ANALYSIS -EXERCISE #1 FLOW CHARTING	-EXERCISE #4 PATIENT REGISTRATION	-EXERCISE #6 CONTROL LOOP	BASIC V ACCESS CONTROL -EXERCISE #8 ACCESS CONTROL	OVERCOMING RESISTANCE TO CHANGE USER TRAINING DBMS FUNDAMENTALS
AF ADP REQUIREMENT, AUTHORIZATION -REQUIREMENTS -EXERCISE #2 MINI-DAR	BASIC II EDITTING	DOCUMENTATION -PROGRAMMER FILE -USER MANUAL -OPERATOR MANUAL	BASIC VI FILES -EXERCISE #9 BUILD/RETRIEVE/ CHANGE A RECORD	-EXERCISE #11 DBMS CRITIQUE
		DINNER		
WHO WANTS TO BE MOSES ?	AF,TRIMIS MEDICAL ADP PROGRAM			

FIGURE 1

2.2 INSTRUCTION CONCEPTS:

The first item of business was to identify clearly the objectives of the entire computer literacy block of instruction. A point of emphasis is that it is not an attempt to make the students programmers or computer specialists, but rather to provide an understanding and appreciation of computer-based systems and the accompanying technical and support aspects of computer system operations Technical hardware and software descriptions and explanations were avoided. Emphasis was placed on the user interface with computer systems and what the user had to know to make the system run.

In the systems analysis area the concept of improvement of a manual system before automating it was stressed. Also the data/information flow (how it is generated and who uses it) is emphasized, especially where the flow crosses normal organizational lines into "nonfriendly" areas. The idea of understanding what problems you are trying to solve before making changes is emphasized as being of paramount importance. In an early volume on the subject of the systems analysis process in a medical environment, Ralph Grams emphasizes this point by repeatedly stating the need to "reconfirm the problem statment" at strategic points in the systems analysis process. (3)

Exercises were developed to provide a practical example of the concepts discussed in the lecture sessions. Two of the exercises--systems analysis and the mini-DAR--were purely manual, but all of the rest required hands-on use of CRT terminals. Since the students were organized into three or four person teams, it was necessary for the instructor to insure that each member of the team spent their share of time on the terminal.

The exercises were designed to build logically upon each other. Only use of an ADT system and use of a DBMS did not require use of the material developed in the previous exercise. The terminal use exercises start with using the editor (exercise 3, figures 2 and 3) and the same basic situation format continued through the exercise on establishing files (exercise 9).

The concept of top down design and stubbing was presented in the context that simply designed, straight forward programs which are well documented (both with good remarks and a detailed programmer's file) are much easier to modify and maintain, especially after the original programmer has departed for places unknown.

The practical subjects of documentation, project management, data privacy, contractor relations hardware maintenance, systems back-up and recovery, overcoming resistance to change, and user training were presented not so much to make the students experts in these areas, but rather to introduce them to concepts, cautions concerning potential problem areas, and references they might find useful.

EXERCISE #3

```
100    REM  ------------------------------------------------------------
110    REM  ****************** PROGRAM   I. D. *******************
120    REM  ------------------------------------------------------------
130    REM
140    REM       PROGRAM FILE # ------------------->   XXXXXX
150    REM       PROGRAM FILE NAME ---------------->   XXXXXX
160    REM       LANGUAGE ------------------------->   XXXXXX
170    REM       COMPUTER ------------------------->   XXXXXX
180    REM       OFFICE SYMBOL -------------------->   XXXXXX
190    REM       ORIGINAL PROGRAMMER -------------->   XXXXXX
200    REM       ORIGINAL DATE COMPLETED ---------->   XXXXXX
210    REM       LAST MAINTENANCE PROGRAMMER ------>   XXXXXX
220    REM       LAST MAINTENANCE DATE ------------>   XXXXXX
230    REM
900    REM  ------------------------------------------------------------
910    REM  ****************** INITIALIZATION *********************
920    REM  ------------------------------------------------------------
950    REM
1000   REM  ------------------------------------------------------------
1010   REM  ****************** ACCESS  CONTROL *******************
1020   REM  ------------------------------------------------------------
1100   REM
1110   REM
3000   REM  ------------------------------------------------------------
3010   REM  ****************** MAIN  MODULE ***********************
3020   REM  ------------------------------------------------------------
3100   REM
3110   REM
4000   REM  ------------------------------------------------------------
4010   REM  ************* ADMISSION  SUBROUTINE *******************
4020   REM  ------------------------------------------------------------
4100   REM
4110   REM
6000   REM  ------------------------------------------------------------
6010   REM  ************* DISCHARGE  SUBROUTINE *******************
6020   REM  ------------------------------------------------------------
6100   REM
6110   REM
8000   REM  ------------------------------------------------------------
8010   REM  ************* TRANSFER  SUBROUTINE ********************
8020   REM  ------------------------------------------------------------
8100   REM
8110   REM
9000   REM  ------------------------------------------------------------
9010   REM  ************* INFORMATION  DESK  SUBROUTINE ************
9020   REM  ------------------------------------------------------------
9100   REM
9110   REM
32000  REM  ------------------------------------------------------------
32010  REM  ****************** EXIT ROUTINE **********************
32020  REM  ------------------------------------------------------------
32030  REM
```

FIGURE 2

EXERCISE #3 INSTRUCTIONS

REQUIRED ------- 1-4
OPTIONAL -------- 5-11

1. USE THE "XFER" COMMAND TO MAKE A COPY OF THE "TEST01.BAS" PROGRAM AND RENAME IT "GROUPX.BAS". THE X IN THE NAME WILL BE YOUR GROUP NUMBER.

2. MAKE THE FOLLOWING CHANGES :

   A.) CHANGE LINE 150 —— XXXXXX TO GROUPX.BAS
   B.) CHANGE LINE 160 —— XXXXXX TO CBASIC
   C.) CHANGE LINE 170 —— XXXXXX TO SD-200
   D.) CHANGE LINE 190 —— XXXXXX TO GROUP X
   E.) CHANGE LINE 200 —— XXXXXX TO TODAY'S DATE

3. MAKE THE FOLLOWING DELETIONS :

   A.) DELETE LINES 1110,3110,4110,6110,8110, AND 9110.

4. GET OUT OF THE EDITOR USING THE "QUIT" COMMAND.

5. USE THE "SPOOL" COMMAND TO GET A PRINTOUT OF YOUR PROGRAM AND SHOW IT TO THE INSTRUCTOR.

6. USE THE "XFER" COMMAND TO MAKE A COPY OF THE "OPTION.BAS" PROGRAM AND RENAME IT THE FIRST SIX LETTERS OF YOUR LAST NAME .BAS .

7. PRACTICE ANY OF EDITOR COMMANDS WE COVERED IN CLASS ON THIS OPTIONAL PROGRAM ONLY. WE WILL BE USING THE PREVIOUS PROGRAM FOR FURTHER EXERCISES SO MAKE NO OTHER CHANGES TO IT.

8. COMPILE THE OPTIONAL PROGRAM.

9. RUN THE OPTIONAL PROGRAM.

10. GET A PRINTOUT OF THE PROGRAM FROM THE PRINTER USING THE "SPOOL" COMMAND.

11. SHOW THE LISTING OF THE OPTIONAL PROGRAM TO THE INSTRUCTOR AND ALSO RUN THE OPTIONAL PROGRAM FOR THE INSTRUCTOR.

FIGURE 3

3. RESULTS TO DATE

3.1 GENERAL:

Student comment about the computer literacy block of instruction has generally been favorable. Although the content of the block has evolved, it has remained rather constant in overall content, changing primarily in modified presentation approach with slightly shifting emphasis from specific techniques to a more general appreciation concept. Our experience has been too limited to make any definitive evaluation. We have made certain preliminary attempts to determine student attitude toward the experience and a brief description of the methods used along with preliminary results follows.

3.2 CLASS CRITIQUE RESULTS:

A critique sheet was developed to determine student perceived value of the block of instruction and securing feedback for areas that needed improvement. Both structured and free form comments were solicited. Student anonymity was incorporated into the critique process--critiques are turned in with no individual identification. Each presentation of the instruction block was critiqued by the students. and the critiques were reviewed and summarized. Results were used as a basis for modifying the block prior to its being presented again.

The structured portion of the critique let the student evaluate the effectiveness of each subject area element of the block on a 1 to 7 scale (low to high effectiveness) with descriptive phrases for each end and middle value. Figure 4 contains three of the structured questions relating to the overall computer literacy block asked on each of the student critiques.

The average evaluation (on a 1 to 7 scale) was computed on the questions listed in Figure 4 each time the computer literacy instruction block was presented. The results are indicated in Figure 5. We find them to be most gratifying.

4. CONCLUSIONS

Computer literacy is essential to operational effectiveness of individuals in our increasingly technical environment. Dr Louis Robinson, Director

CRITIQUE QUESTIONS ON OVERALL EFFECTIVENESS
OF THE COMPUTER LITERACY BLOCK
OF TESTING

HOW DO YOU RATE THE OVERALL VALUE OF THIS TYPE OF EXPERIENCE AS A TEACHING VEHICLE FOR MEDICAL COMPUTER LITERACY ?

POOR GOOD EXCEPTIONALLY WELL SUITED

1 2 3 4 5 6 7

HOW DO YOU RATE THE APPROPRIATENESS OF THIS TYPE OF EXPERIENCE TO COMPUTER LITERACY TRAINING ?

NOT APPROPRIATE APPROPRIATE VERY HIGHLY APPROPRIATE

1 2 3 4 5 6 7

DO YOU RECOMMEND THAT THIS TYPE OF EXPERIENCE BE PROVIDED FUTURE MSC CLASSES ?

NO POSSIBLY IF PROPERLY MODIFIED YES

1 2 3 4 5 6 7

FIGURE 4

AVERAGE RESULTS OF SELECTED
STUDENT CRITIQUE QUESTIONS
(SCALE OF 1-7)

SESSION DATE	NUMBER OF STUDENT RESPONSES	INSTRUCTION BLOCK AS A VEHICLE TO TEACH COMPUTER LITERACY	APPROPRIATENESS OF INSTRUCTION BLOCK FOR COMPUTER LITERACY	RECOMMEND INSTRUCTION BLOCK ON COMPUTER LITERACY FOR OTHER MSC'S
3 JUNE-6 JUNE 1981	12	5.5	5.6	6.3
20 JULY-24 JULY 1981	44	5.4	5.1	6.3
19 OCT-23 OCT 1981	16	5.9	6.1	6.6
25 JAN-29 JAN 1982	28	5.4	5.1	5.9

FIGURE 5

of Scientific Computing for IBM's Data Processing Division, stated in an interview by Output that by 1985 seventy percent of the nation's work force will have to know how to interact with a computer to hold a job.(4) This is fast becoming a reality in the Air Force Medical Service.

Both educational and operational institutions must co-operate to provide this future human resource so essential to increased productivity. The co-operative arrangement among the Air Force School of Health Care Sciences, the Wilford Hall USAF Medical Center, and the Air Force Medical Service Center serves as an example of how this can be achieved. Generalized conclusions drawn from our experience could be summarized as follows:

- Educational institutions should have the mission to teach computer literacy and integrate it into its curriculum so that it is presented as a normal part of the job rather than for "data automation specialists."

- Operational institutions should assist educational institutions in developing computer literacy programs, providing both technical and practical input along with instructors and equipment when necessary.

- Computer literacy education should be practical so that the student can relate to it. Hands-on use of computer systems is essential to reinforce theoretical concepts in a real-world environment.

5. REFERENCES

1. Andrew R. Molnar, "The Next Great Crisis In American Education: Computer Literacy," Association for Educational Data Systems Journal, Fall 1978 (Vol. 12, No. 1), p.13.

2. See Hillel Segal, "Expandability Key Feature of SD-200," Computerworld, 30 November 1981, p. 28, for results of a standardized benchmark test performed by the Association of Computer Users on the SD-200 microcomputer.

3. Ralph R. Grams, Problem Solving, Systems Analysis, and Medicine, (Charles Thomas, Springfield, IL) 1972, pp. 53-60.

4. ______________________________
"Spotlight: Louis Robinson," Output, February 1981, pp. 81-82.

DEVELOPING A COMPUTER
CURRICULUM FOR A
COLLEGE OF EDUCATION

John F. Vinsonhaler
Professor, College of
Education
Mich. State University
(517) 353-6471

I. INTRODUCTION

I first began thinking about a computer curriculum for our college in 1978. At that time, many of our faculty were teaching bits and pieces of computer lore in independent unrelated courses. Further, a good collection of hardware resources were scattered throughout the the various research projects in the college. It seemed to me, given the rapidly growing need for computer training in education, that we might create a valuable instructional resource by simply combining and coordinating our existing resources. As it turned out, the project did not prove all that simple or easy. This paper is a summary of my colleagues'* experiences, both successful and unsuccessful, in creating such a curriculum. Our adventures may or may not be representative of those others who make this pilgrimage.

II. THE COMPUTER ADVISORY COMMITTEE

Studies of the adoption of innovations indicate that success is most likely when the innovation is backed by leaders of an organization. With this in mind, my first step was to propose to the Associate Dean of my college (in charge of Research and Development) the idea of a computer advisory committee to be chaired by the Associate Dean and to help better manage college computer decisions. My suggestion was accepted. The fact that the Associate Dean was research oriented, and intensely interested in computers certainly helped my case. The committee was formed ad hoc to include the five faculty in the college who were serious users of computers. One graduate student was also included to advise on student concerns.

The first two items presented to the committee were: 1) the concept of a centralized computer resource center and 2) the development of a series of courses on computers for the college. These two initial items have become permanent items on the agenda of the committee. The committee has also proved to be useful in evaluating and planning many other computer related activities in the college.

III. THE COMPUTER ACCESS CENTER

It has been said that no item can be purchased by the Pentagon until the weight of the paperwork equals the weight of the item, i.e., battle tank, jet fighter, etc. Most university committees seem to follow this standard. Happily, the Computer Advisory Committee--thanks to a strong chairman and an action oriented majority-- proved the exception.

Almost immediately the committee approved the establishment of a Computer Access Center (CAC) as a unit in an existing Instructional Materials Center. To give the fledgling center a start, I committed a terminal and two microcomputers (DEC PDP-8 systems) obtained through a grant from the federal government. The college matched my contribution with microcomputers and some software. The total initial cost to the college was about $2,500. The center was operated as a part of the Instructional Materials Center with part-time student assistants. After the center was established, other researchers began to contribute hardware and the college reciprocated by purchasing microcomputers and terminal/printers. At present, even researchers outside the college are contributing hardware.

---

*I would especially like to acknowledge those Michigan State University faculty whose help has been critical: Prof. Castelle Gentry, Prof. Andrew Porter, Prof. Robert Craig, Prof. William Schmidt, Prof. Lee Dean, Prof. Don Meaders, and Prof. Clinton Cobb.-- all of the College of Education at Michigan State University.

The advantage from the research project point of view is that the CAC is responsible for supervision and maintenance of hardware, while the research projects retain priority usage.

Presently, our Computer Access Center provides; (1) access to mainframes (Amdahl 470, CDC 750) via multiple terminals (Decwriters, DIABLO TYPE II), minicomputers, and microcomputers (e.g., APPLEs, ATARIs, a TRS-80, a PET Commodore, and DEC PDP 8's).

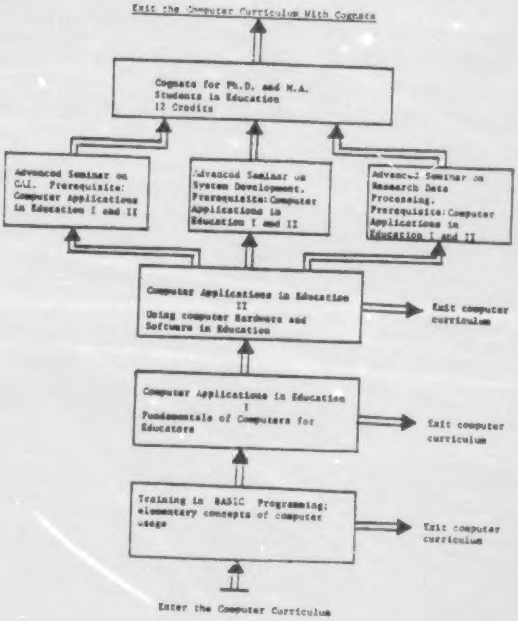
In my opinion, an equivalent of our CAC is a major requirement for the development of a computer curriculum for any College of Education. Without such a center, the curriculum lacks a basis for the accummulation of hardware and software critical to adequately train education students. One of the most important maneuvers we employed in funding the CAC software/hardware deserves mention. Shortly after the center was established, I prepared a prioritized plan for hardware and software acquisition for the present and the next fiscal year. As is well known to most administrators, university budgets are often subject to last minute surpluses. Our acquisition plan made use of this fact and won vitally needed resources for the center.

The major problems we have encountered with our access center concern its organization and management. The center was established as an independent organization without any formal ties to the instructors of our computer courses. As a result, the center is not very responsive to its major source of users, i.e., the college computer curriculum. In this regard we are especially lacking a sufficient acquisition of high quality software packages for CAI.

IV. THE COMPUTER CURRICULUM

Our curriculum has been developed in several stages which do not follow any rational order. We began with two basic courses for graduate students because these were the courses most acceptable to our administrators. We are now expanding the curriculum both downward (an undergraduate course in BASIC) and upward (advanced seminars in CAI, research, and administrative applications). An overview of the general structure of our computer curriculum may be summarized in terms of the components in figure 1.

FIGURE 1
A College of Education Computer Curriculum


Exit the Computer Curriculum With Cognate
Cognate for Ph.D. and M.A. Students in Education 12 Credits
Advanced Seminar on CAI. Prerequisite: Computer Applications in Education I and II
Advanced Seminar on System Development. Prerequisite: Computer Applications in Education I and II
Advanced Seminar on Research Data Processing. Prerequisite: Computer Applications in Education I and II
Computer Applications in Education II Using computer Hardware and Software in Education
Exit computer curriculum
Computer Applications in Education I Fundamentals of Computers for Educators
Exit computer curriculum
Training in BASIC Programming; elementary concepts of computer usage
Exit computer curriculum
Enter the Computer Curriculum

A. Training in BASIC Programming (3 credits)

During the first two years of course development, graduate students were permitted to learn BASIC programming as a part of the graduate level sequence. This led to two major difficulties. First, learning BASIC in addition to everything else that had to be learned tended to overload the graduate courses. Second, the Computer Science Department objected to the College of Education offering graduate credit for what was being taught by Computer Science as an undergraduate course. Therefore, a prerequisite was introduced for our first graduate course (ED 897A): students must have a working knowledge of BASIC. The prerequisite can be met in three ways: 1) completing the Computer Science (CPS) 115, "Introduction to Programming," course; 2) Completing the ATARI, "Introduction to Programming", CAI sequence in the Computer Access Center; or 3) Instructor's approval, via a test on BASIC.

All of these methods fail to meet an obvious need of providing instruction in BASIC and CAI software for teachers who already have advanced degrees but who know almost nothing about computers. To meet this need we are now working on a new type of BASIC course devoted to teaching classroom teachers how to teach computer literacy in the classroom using BASIC and other software systems.

B. Computer Applications in Education I (3 credits)

Our first course was created in an environment of change. In the face of declining budgets, our new Dean and her staff were attempting to bring about a transition to a new college which would be quantitatively smaller in number but qualitatively better in scholarship and research. In accordance with this mission to seek better quality, the Associate Dean wanted a course with high academic respectability. On the other hand, I and my colleagues were well aware of the limited background of many education graduate students in systematic thinking. Our first graduate course is a compromise of these two needs. The contents of our first course are summarized in Figure 2. The three major components require comment. About one third of the course is devoted to explaining the computer in terms of: 1) its role as one element of the information processing system; 2) its evolution as the most effective intellectual tool yet developed; and 3) its inner workings in terms of storage, processing, and control. The first two of the above components are common to most college undergraduate text books on computer literacy. The third is not generally taught to non-specialists.

Figure 2: The First Graduate Course on Computers:

- a. De-Mystifying Computers
  - 1) What are Computers?
    - a) Information Processing Systems
    - b) Components
      - i) Computer and programs
      - ii) People and procedures
      - iii) Data
  - 2) How did Computers Evolve?
    - a) Evolution of computer hardware
    - b) Evolution of computer software
    - c) Evolution of the computer industry
  - 3) How did CAI Evolve?
    - a) Early research beginnings
    - b) Halycon years 1960-70
    - c) Retrenchment and new beginnings .
  - 4) How do Computers Work?
    - a) Input/Output (I/O)
    - b) Storage
      - i) Numbers
      - ii) Strings
    - c) Processing
      - i) Logical
      - ii) Arithmetic
    - d) Control
      - i) Widget Factory Model
      - ii) MARK I Model (Automatic Sequencing, Direct Addressing) (Grades of Less Than 3.0)
      - iii) MARK II Model (Branching and Address Modification) (Grade above 3.0)
- b. All About Computer Hardware and Software
  - 1) What is Computer Hardware?
    - a) Input/Output (I/O)
    - b) Mass Storage
    - c) Mainframes
    - d) Minicomputers
    - e) Microcomputers
  - 2) Midterm
  - 3) What is Computer Software?
    - a) Operating Systems
    - b) Application Systems
    - c) Programming Systems
  - 4) How do you Program?
    - a) Languages: BASIC
    - b) Flow Chart Programming
    - c) Structured Programming
- c. Applying Computers in Instruction
  - 1) What has been done?
  - 2) How do you select hardware and software for CAI?
  - 3) How do you develop CAI Applications?
- d. FINAL EXAM

I chose to teach how computers actually work in terms of storage, control, and processing registers on the basis of two arguments. First, the ACM curriculum for teachers includes the goal of understanding computer functioning at the assembly language level. Second, how computers really work is not all that hard to learn. Using CAI materials developed in the college we have successfully taught all of our students the concepts of Automatic Sequencing and most of our students the concepts of Branching and Address Modification (e.g., see Ralston & Meek, 1976, pages 8-18). Our criterion of understanding is the ability to: 1) describe the execution of a pseudo-assembly language program; and 2) to modify the program to handle different data sets. The midterm exam option allows the student to limit himself/herself to Automatic Sequencing or to attempt questions required for Address Modification and Branching.

The second third of the course is devoted to computer hardware and software. The hardware components used are those usually taught in introductory courses (types of CPU, Memory, and I/O) but augmented with problem sets that require the student to make practical judgments about the capabilities and costs of various hardware configurations and their suitability for use in educational applications.

In studying software, students are taught both the traditional flow chart and the structured programming approaches. All students must complete a programming task. Programming tasks are offered at two levels: 1) traditional flow charted programming; and 2) structured programming.

The final third of the course is devoted to "crash" training for dealing with Computer Aided Instruction (CAI) and Computer Literacy for classroom teaching. The last component of the class includes: 1) a review of CAI from the literature; 2) a "CAI Primer" to aid in initiating Computer Literacy in a typical classroom and 3) a simulated experience in the selection of CAI hardware and software.

The course materials include:

1) A TEXTBOOK (Vinsonhaler, et al. 1982);
2) Class materials (Gentry, Wolverton, & Vinsonhaler, 1982) and
3) Video cassette and CAI components (Wolverton, 1982).

C. Computer Applications in Education II (3 credits)

The second graduate course is concerned with applying computers to solve problems in Instruction, Administration, and Research. About one third of the course is devoted to the type of applications described in the literature. Students are taught 1) a method of classifying applications; 2) examples of major application projects for each type; and 3) to evaluate each type of application in terms of present research findings.

Two thirds of the second course is devoted to familiarization with commonly available computer hardware and software. Each student is assigned the task of preparing an instructional module which will teach the basic operation of a software package. All students are assigned the task of using previously developed instructional modules to acquaint themselves with software for: 1) Administration (e.g., word processing, accounting, and report generation); 2) Instruction (e.g., terminal-oriented data processing, and general purpose author languages); and 3) research (e.g., data base management, and statistical data processing).

D. Advanced Graduate Courses (3 credits each; prerequisite Computer Applications I and II)

The advanced seminars in the college are revisions of courses previously offered by the college faculty. We are emphasizing three major areas in the development of these advanced courses.

The CAI Seminar is designed to allow students to study in detail: 1) Computer Literacy (developing instructional modules for teaching about computers) and 2) CAI (using computers as an aid in instruction). The Research Data Processing Seminar is designed to train students in the use of computers for data base management and statistical analysis. The Systems Development Seminar is designed to provide students with an experience in the development of a computer-based system for Instruction, Administration, or Research.

E. The Computer Cognate

The cognate offered to Ph.D. and MA students in our college is any combination of the two basic graduate courses (6 units) and six additional units taken from the advanced graduate seminars.

F. Success and Problems

Given the resources of our college, the curriculum I have described could not have been developed without the participation of existing faculty, some of whom initially lacked any prior training in computers. The Associate Dean and I felt that it was important that the curriculum be a team effort rather than a set of courses developed and controlled by a single faculty member. Much use was made in the first two courses of video cassette lectures and word processed written materials (using TEXTFORM on the Amdahl 470) and CAI programs. These media and computer-based teaching aids have permitted instructors to teach our introductory course after only one term as an intern -- without any observable loss in student learning.

On the negative side, it has been very difficult to "bring in the mavericks." Some of our faculty, with the most valuable computer knowledge, have been the most difficult to recruit to our project. Often the problem has been outside committments of faculty with valuable skills. The inalienable right of faculty to distrust any move to do anything to his/her courses has also been a problem.

IV Conclusions

I have emerged from the experience of developing an educational computer curriculum with only one firm opinion. Such a curriculum can be developed in almost any college of education given time, committment, and above all, patience. Three techniques have greatly aided us in developing such a curriculum. They are stated below in terms of imperatives.

First, establish a formal mechanism (e.g., a committee) to marshall support for computer developments within the Dean's "inner circle" of advisors. At this point, at least one supportive high level administrator is critical to the whole enterprise.

Second, solicit support from the administration and the faculty in establishing a college computer center using existing computer hardware and software, e.g., from research projects. Request matching funds from the college and/or the provost's office.

Third, patiently build, course by course, a cognate on computers. Minimize costs by using existing college personnel whenever possible. In this regard, some faculty long written off as "deadwood" may prove to be valuable resources-- given a chance to serve a computer internship.

Finally, after you have managed to do all this, do not expect canonization. After all, if you have done your job really well, the college administration and faculty will probably be convinced that the computer curriculum was mainly their idea anyway.

V References

Vinsonhaler, J. Handbook of Computer Applications in Education. Michigan State University, 1980.

Gentry, C., Wolverton., & Vinsonhaler, J. Computer Applications in Education I: Student's Syllabus. Michigan State University, 1982.

Ralston, A. & Meek, C.L. Encyclopedia of Computer Science. Mason/Charler Publishers, Inc., 1976.

Wolverton, W. Video and Computer Based Teaching Aids for Computer Applications. Michigan State University, 1982.

Special Session

IMPLEMENTING MODEL CURRICULA

Chared by: Daniel V. Goulet
University of Wisconsin-Stevens Point

ABSTRACT

The computing profession has been concerned with the educational preparation of entry level professionals, and have published a variety of curriculum studies designed to assist colleges and universities in curriculum preparation. The most notable studies are: ACM's curriculum 78, IEEE's A Curriculum in Computer Science and Engineering, ACM/s Educational Programs in Information Systems, DPMA's Model Curriculum, Pittsburgh Large User Group Education Committee's Model DP Curriculum, and CUPM's Report of Subpanel on Computer Science. Each proposal specifies a set of courses and content, and a sequencing through the course lists. Some specify goals and terminal objectives for students completing the course structures. All address the curricular problem from the professional activity point of view. None address the problem from the educational setting point of view.

The thesis of this panel is that all curricula exists in a particular educational environment and must be tailored to that environment. The panel will discuss a methodology for examining the global problem--model curricula and educational environment, and case studies which suggest implementation strategies, tradeoffs, and compromises.

Participants

John Beidler
University of Scranton

Thomas Ho
Purdue University

Project Presentation

APPLICATIONS AT ABOUT THE HIGH SCHOOL LEVEL

Chaired by: Richard Close
U.S. Coast Guard Academy

Karen Billings
Warren R. Brown
James T. Fey
Charles D. Friesen
Chuck Grady
Creig Humes
Lloyd A. Jerauld
Jim Kelly
Robert Kraft
Antonia Stone
Alisha Whipple
Kara Gae Wilson

ABSTRACT

Project C.A.I.S.H.-Computer Assisted Instruction & Support for the Handicapped

Warren R. Brown, Project Manager, School Board of Sarasota County, 3450 Cocio Road, Sarasota, Florida 33580

Sarasota County has started a new program, COMPUTER ASSISTED INSTRUCTION & SUPPORT FOR THE HANDICAPPED, under the Education for ALL Handicapped Children Act, 1975 (Public Law 94-142, Title VI-B).

The project addresses the instructional problems presented by students with handicaps which in hibit their ability to communicate. A major difficulty experienced by teachers of handicapped students is finding methods through which students may receive and respond to instruction, i.e. answer questions, practice skills which have been introduced, and demonstrate mastering of skills. The project uses systematic methods to identify and provide microcomputer-based devices and materials which will meet the needs of students with specific handicaps.

The program is based on a four-part rationale:

(1) Adaptive devices are required to facilitate communication for orthopedically impaired students during instruction.
(2) Microcomputers can be interfaced with adaptive devices to create a highly interactive learning environment for orthopedically impaired students.
(3) Computer-based learning methods are effective with, and motivational to all types of handicapped students, and can maximize teacher time.
(4) The use of systematic methods can result in optimally useful instructional methods and products for all types of handicapped students.

The project is supplying handicapped students with a means of communication and with direct instructional support. Microcomputers are used to provide a communication link between students, teachers and instructional materials. A three year plan is underway to develop and adapt a wide range of curriculum materials for use by handicapped students, including deaf, emotionally disturbed, learning disabled, mentally handicapped and orthopedically impaired students.

Through the use of microcomputers, even students with extremely limited fine and gross motor ability are able to respond and interact, using adaptive devices to compensate for their particular disabilities. Because their need for a means of communication is so acute, the first year of the project is directed toward the orthopedically impaired students. Devices and software interfaces for microcomputers have been acquired and developed for individual students. Student's Individual Educational Plans (IEP's) are used to determine individual needs as a guide for the selection and/or development of instructional courseware.

In summary, the project interfaces proven, effective instructional and technical capabilities fo the microcomputer with adaptive devices which have proved successful in maximizing the communications abilities of students and in delivering individualized, adaptive instruction in many areas including math, language, arts, science, social studies, etc. The resulting systematically designed models are providing an environment responsive to the individual instructional needs of handicapped students.

For further information, contact: Warren R. Brown, Project Manager (813) 355-3567.

ABSTRACT

Implications for Secondary School Mathematics Curricula

James T. Fey, Div. of Mathematical and Physical Sciences and Engineering, University of Maryland, College Park, Maryland 20742

Existing and projected microelectronic technology permit machine performance of almost all formal manipulations that students now learn in high school mathematics. However, to date school curricula have undergone almost no broad change to reflect emergence of powerful computer based work environments in science and mathematics.

Experience with attempted innovations of the "new math" era suggest that for appropriate change to occur as rapidly and smoothly as possible, work must begin soon on three basic questions:

1. What are the best possible projections of mathematical software likely to be available for use in doing high school mathematics by 1990?
2. What mathematical skills and understandings will students need in order to use the computing power effectively?
3. How might appropriate change in school mathematics be initiated, given the complex interrellation of curriculum expectations by students, parents, teachers, and college faculty?

The project will conduct exploratory investigations into each question, involving all impacted parties in probing analysis of the possibilities.

ABSTRACT

The Teacher Oriented Computer System (TOCS) Approach to Remedial Mathematics

Lloyd A. Jerauld, Mathematics Department, Jamesville Dewitt High School, Dewitt, New York 13214

TOCS carries many of the attributes of the existing CAI math programs, using the computer to carry out subservient duties, but is powerfully different in that it is based on the following principles.

> THERE ARE CERTAIN PEDAGOGICAL TASKS AND DECISIONS THAT ARE NOT CONDUCTIVE TO COMPUTER ANALYSIS AND CAN BE MORE EFFECTIVELY CARRIED OUT BY A TEACHER.

The application of the principle allows for tutorials and worksheets to be generated and corrected by the computer (along with full record keeping facilities), but the responsibility of the content design is left to the teacher. Each tutorial may be constructed in sections, possibly containing mixed problems, similar problems, easy, hard, etc., depending on the teachers analysis of the needs of the class or the individual student for which it is designed.

The interactive tutorials indicate errors and allow for corrections with repeated mistakes. The students are led through the problem step-by-step by having them respond to key questions. If a student demonstrates an inability to comprehend the process at any step, the basic TOCS principle is applied: the computer instructs the student to get help from the teacher and the keyboard is locked from further use until the teacher has diagnosed the problem.

The decision of advancement through the assignments is also mandatorily a teacher decision. After each assignment, the computer calculates the displays the relevant statistics, but it is the teacher who exercises the option of advancing the student, having the assignment repeated, or substituting a different assignment specialy designed for that student.

The TOCS is written for the TRS-80 and is being used in several classes containing 6 to 8 high school students to prepare them for the New York State Regents Competency Test in Mathematics. The presentation will include a computer demonstration of features such as designing a tutorial, computer interaction during tutorials, and built in answer analysis which discriminates careless typing errors. Hand outs will be available showing examples of the included demonstration, worksheets, and print-outs of student records.

ABSTRACT

Cross Lingual Mathematics Information System

Jim Kelly, 1000 Grove, Evanston, Illinois 60201

With a random sequential file consisting of 1000 records, each containg English, Spanish, and French equivalents of commonly used conceptual, operational and notational elementary and secondary school mathematic terms; a simple but useful data base has been developed. The information system will serve the classroom teacher, curriculum designer and the mathematics educational community in general.

The project is in three phases. The first phase consists of a program which allows the user to do direct searches or category searches for the language equivalent to any of the 1000 terms. The software was written for an Apple II (48K) with two disk drives and monitor. The second phase is the accumulation of content information for a basic information retrieval system. The content of 152 elementary and secondary school books from five countries have been compiled into a 24,000 record file. Each of the 19 byte records shows, through special coding where and how the mathematics terms are used in the book. The software developed, accesses the terms file of phase one, selected portions of the content file and a small 125 byte bibliographic information file to give the user cross lingual information either by term, book or grade level. Work on phase three is just beginning. It will consist of multiple language scope and sequence creating, comparative methods on examining curriculums and curriculum designing simulations. A printer and hard disk will be added to the Apple II.

ABSTRACT

COMPU-GRADE: Teacher's Gradebook Programs

Charles D. Friesen, Lincoln High School, 22nd and J, Lincoln, NE 68510 and Robert G. Kraft, 1900 South 25th, Lincoln, NE 68502

COMPU-GRADE is a comprehensive record-keeping system for educators. Designed and field-tested by classroom teachers over the past two years, the programs free professionals to spend time with students -- not records!

In this twelve program package, the traditional functions of storing and computing grades are enhanced by a package that permits sophisticated statistical analysis if desired. The report card component provides instant feedback for students and their parents.

Teachers or their assistants can easily enter student scores, weight individual items and get a sum or average in seconds. The socres for a single student or an entire class can be safely stored until grade reports are printed.

HOW IS COMPU-GRADE USED BY CLASSROOM TEACHERS?

To store and average (or sum) student scores.

To display the range and distribution of socres for a single test or a single student.

To analyze scores with a complete statistical package.

To make a class printout of scores to post in the classroom.

To print individual report cards with persomal messages.

It is easy to proofread and edit, as well as to update files with new scores or student names. The program makes back-up copies to prevent loss of data and insure safety. Student numbers may also be used in place of names for anonymity.

TEACHERS SAY COMP-GRADE WORKS!

Report cards are completed in minutes.

Students receive immediate status reports.

Individual reports list grades and missing work.

The statistical package allows analysis of mean, median, mode, standard deviation, variance, range, frequenc distributions, and z-scores for each column in the gradebook and for the averages or sums.

No previous computer experiences are necessary.

This two year project in software development has resulted in an 18 page manual and diskette for use on either the Apple II or the TRS-80 microcomputer. These programs are being used extensively by the Lincoln Public Schools at the present time.

ABSTRACT

Students Teaching Students About Computers

Chuck Grady, Creig Humes, Alisha Whipple, 2027 Cunningham Drive, Apt 201, Hampton, Virginia 23666

This program got started when the computer systems at Robert E. Lee Elementary School in Hampton, Virginia were taken away because of budget cuts. We found out about an opening for teaching these children through our own Computer Math teacher, Mrs. Sheila Rogers, at Hampton High School. She asked for volunteers. We volunteered without knowing what wer were really getting into. We had very little computer experience and were not sure of how to go about the task of teaching others.

The first thing we taught was how to get on the computer terminals by turning the terminal on, then by dialing the number on the phone of the building the computer is in and attaching the phone to the acoustical coupler then typing their code number and the language they are going to use. Then to let the students get used to using the terminal keyboard we let them play some games.

The second week of class we were ready to start serious work. We wanted to find a way to introduce flowcharting that would make it interesting. Mrs. Rogers found an article by Marilyn Burns entitled "Getting Kids Ready for Computer Thinking". In this article, Mrs. Burns tells how she used a little sketch involving a peanut butter and jelly sandwich to teach children the way a computer works. We decided this was a good way to get the interest of the children, but we weren't quite sure how it would turn out. We were pleased with the success of this introduction and decided to get into flowcharting problems.

Unfortunately, the students did not catch on to flowcharting very well. We tried going through a simple flowchart step by step in front of the class. We tried explaining the process individually to each student. However, we just couldn't get the message across. It was then that we decided to put flowcharting up for a while and start BASIC programming.

So far, we are pleased with the amount

the students are learning and their cooperation. Even though their attention spans are limited at times. We have enjoyed our class and from it we have learned some of the difficulties with teaching others what we know.

ABSTRACT

The Implementation and Evaluation of Microcomputer Use By Incarcerated Juveniles at the Spofford Juvenile Center

Antonia Stone, Executive Director, Playing to Win, Inc., New York, NY and Karen Billings, Project Evaluatior, Columbia Teachers College, New York, NY

The Spofford Computer Project was designed by Playing-to-Win, Inc. to provide an opportunity for institutionalized juveniles to benefit from educationally oriented microcomputer games. In this project twenty-four microcomputers were installed and used in Spofford's learning center classrooms, dormitories, and recreational facilities and the Family Court detention rooms in four of New York's five boroughs. The project included supervising the installation of the computers, training of institution personnel in their operation, collection and development of software correlated with Spofford's educational program, and supervision of initial phases of use by institution staff and inmates. This five-month phase ran from July 1 to November 30, 1981.

The project grew out of a previously implemented program at a Manhattan rehabilitation agency for ex-offenders, called the Fortune Society. Some of the ex-offenders in that computer program became trainers in the Spofford project. The Spofford Juvenile Center is a secure facility for 10-17 year olds and is operated by New York City's Department of Juvenile Justice. Spofford is located in the South Bronx and houses approximately 220 juvenile deliquents and offenders.

An evaluation of the period was also conducted to study the effect of microcomputers on the Spofford community. The major part of the study focused on the different concerns and computer uses by the Learning Center staff as well as on the cognitive and affective changes for the students. Teacher interviews, questionnaires and observations were conducted before, during and after the initial installation phase. Student data was compiled with pre- and post-testing, observations of actual computer use and interviews. The evaluation effort ran from July 1, 1981 to March 1, 1982.

New York Community Trust financed the evaluation and, together with the City of New York, provided funding for the project.

ABSTRACT

The Teachers Desk of Tomorrow-Today

Georgian Guse, Math/Computer Instructor
Kara Gae Wilson, EdD, Assistant Principal, Broken Arrow Senior High School, Broken Arrow, Oklahoma 74012

This presentation will demonstrate a variety of computer applications for administration and classroom management currently being used in the third largest high school in Oklahoma. While the major portion of the demonstration will center on software features of The Radio Shack TRS-80 Model III, attention will also be given to the advantages and disadvantages of batch processing for scheduling and grading through a large commercial computer service.

This particular demonstration is one part of a six week staff development course entitled "Computer Literacy for Educators" for which Mrs. Guse' is district coordinator.

Program Sequence:

Administrative Uses of Computer Technology:
Radio Shack and Westinghouse
Classroom Uses
Test Creation and Stencil Production
Test Analysis and Record Keeping
Progress Reports
Lesson Plans for Semester
Newspaper by Computer
Programs, Tickets
Personal Uses - Letters, Cards, Etc.
Curriculum Enhancement:
Teacher-Created Software (Astronomy)

We will provide the equipment for the demonstration: TRS-80 Model III; Radio Shack Line Printer IV; Chatsworth Data Corporation Card Reader

RATING THE MAJOR COMPUTING PERIODICALS ON READABILITY

Ronald S. Lemos
Department of Business Information Systems
California State University
Los Angeles, California 90032

Abstract

Comparative readability of the ten major computing periodicals is analyzed. A program to derive the Flesch Reading Ease Index is used as the measure of text readability. This formula estimates readability based upon sentence length (number of words) and word length (number of syllables). Sixty text samples of approximately 100 words each are systematically selected from each periodical. The periodicals represent both "technical" and "management" oriented publications.

Results showed all periodicals to be categorized as "very difficult" and "difficult" reading characteristic of scientific and academic journals, respectively. IBM Systems Journal had the lowest readability index, while Commumications of the ACM had the highest. Thirteen significant pairwise differences in index means are identified. Variability was also found to be significant between issues and articles within magazines. No significant differences were found between the four "technical" and the six "management" periodicals.

This study shows the feasibility of using readability indexes to estimate readability levels of computing periodicals. These results can be used by readers, writers, and publishers in focusing on a specific area of effective presentation of computing information.

Introduction

An imporant challenge in any profession is maintaining one's currency in the field. This is particularly important in computing. New concepts, changes, and developments occur so quickly that computing professionals must make special efforts to remain current. There are several ways to keep one's knowledge up-to-date. These include:

- Discussions and interactions with other professionals;
- Books and manuals;
- College/University courses;
- In-house training;
- Seminars and conferences; and
- Professional journals/periodicals.

The latter are particularly important sources of organized and relatively current information. However, one problem is dealing with the sheer number of different periodicals published. The Quarterly Bibliography of Computing the Data Processing lists over 120 publications. About 60% deal with computing as their main theme with the rest having their main focus in another discipline (e.g. Management, Electronics, Business). In addition, many new periodicals continue to appear each year.

Factors that influence which periodicals get read include:

- availability of the periodical;
- cost of periodical;
- awareness of the periodical's existence;
- articles of particular interest in a periodical;
- the "snob value" of the periodical; and
- perceived readability of the periodical.

The last factor is very important. A periodical that is not felt to be readable will probably not get read by a person with significant time constraints. Therefore, it stands to reason that publishers of computing periodicals should be concerned with the readability of their products. Interest in readability can include estimates of difficulty levels, comparisons of readability measures between different issues or articles, and consistency within articles.

This study involves the measurement of the readability of the major computing periodicals. The Flesch Reading Ease Index (4) is used as the measurement tool and is described in the next section.

The periodicals are measured for comparative readability, differences between technical and management oriented publications, and variability. This study provides empirical evidence on readability levels which can serve as a basis for increased attention to this important area.

Methodology of the Study

Sample

The sample for this study consisted of the top ten computing periodicals in terms of reported circulation. Circulation data was derived from two sources (2, 12). No distinction was made between controlled circulation (free) and paid circulation figures. Table 2 presents the circulation rankings.

The STAR Program

The ten computing periodicals were measured for comparative readability using STAR. STAR is a computer program that was developed by the Service Research Group at General Motors to analyze the readability of shop manuals (5). Essentially, the program computerizes the use of the Flesch Readability Index (R.E.) and a derived Dale Index (Dale-Chall score) that transforms R.E. scores into grade-in-school equivalents.

Passages to be analyzed are entered in the computer (on line) and stored as text data files. The program is then used to process these files and provide the following information on each passage (see Table 3):

printing of the passage;
words (and frequencies) having three or more syllables;
number of sentences;
number of syllables;
average sentence length;
average syllables per word;
Flesch Readability Index (R.E.); and
reading grade level (derived from Flesch Index).

Table 2

Computing Periodicals: Circulation Ranking

Rank	Name	Estimated Circulation	Primary Audience
1	Infosystems	110,000	Management
2	Datamation	101,000	Management
3	Computer Decisions	85,000	Management
4	Mini-Micro Systems	82,000	Management
5	Computerworld	67,000	Management
6	IBM Systems Journal	57,000	Technical
7	Communications of the ACM	35,000	Technical
8	Data Management	30,000	Management
9	Computer	24,500	Technical
10	Computer Journal	20,000	Technical

Table 3

Example of STAR Output*

```
-------------------------------------------------------------------------------

NAME OF SAMPLE TEXT          N021            81/12/06.
-------------------------------------------------------------------------------

TEXT

ALTHOUGH THE DATA BASE APPROACH BRINGS MANY ADVANTAGES, IT ALSO CREATES
NEW, OR MORE INTENSE, PROBLEMS IN TERMS OF SECURITY. IN THE TRADITIONAL
APPROACH, EACH APPLICATION SYSTEM HAS ITS OWN FILES. A LIMITED AMOUNT OF
DATA IS SHARED BY PASSING TAILORED FILES FROM ONE SYSTEM TO ANOTHER. IN A
DATA BASE ENVIRONMENT, MANY USERS WITH DIFFERING SECURITY REQUIREMENTS,
SOME ON LINE, HAVE ACCESS TO A COMMON POOL OF DATA OF VARYING DEGREES OF
SENSITIVITY. ACCESS TO THE DATA MUST BE CONTROLLED, THEREFORE, TO ENSURE
THAT PRIVACY REQUIREMENTS ARE SATISFIED, LEAKAGE OF ENTERPRISE SECRETS IS
PREVENTED, AND THE POSSIBILITY OF ACCIDENTAL OR MALICIOUS CORRUPTION OF
DATA IS MINIMIZED. HOWEVER, THE CENTRALIZED NATURE OF DATA BASES ALSO
PROVIDES A BASIS FOR SYSTEM-WIDE SOLUTIONS.
-------------------------------------------------------------------------------

HI-CAL WORDS

 1 ACCIDENTAL  1 ADVANTAGES   1 ANOTHER        1 APPLICATION  1 CENTRALIZED
 1 CONTROLLED  1 CORRUPTION   1 DIFFERING      1 ENTERPRISE   1 ENVIRONMENT
 1 HOWEVER     1 LIMITED      1 MALICIOUS      1 MINIMIZED    1 POSSIBILITY
 1 PREVENTED   1 PRIVACY      2 REQUIREMENTS   1 SATISFIED    2 SECURITY
 1 SENSITIVITY 1 SOLUTIONS    1 SYSTEM-WIDE    1 THEREFORE    1 TRADITIONAL
-------------------------------------------------------------------------------

SUMMARY AND CALCULATIONS

NUMBER OF SENTENCES=   6.0  AVERAGE SENTENCE LENGTH=  20.2
    NUMBER OF WORDS= 121.0  AVE. SYLLABLES PER WORD=   1.8
NUMBER OF SYLLABLES= 221.0            FLESCH INDEX=  31.8
             READING GRADE LEVEL=  15.7
-------------------------------------------------------------------------------
```

*Source: Wood, C. et al. Data base security: requirements policies, and models. IBM Systems Journal, No. 2, 1980.

The researcher also has the option of requesting a numerical summary table of all passages (see Table 4).

Table 4

Example of STAR
Summary

```
------------------------------------------------------------------------

                        SUMMARY OF DATA

                                          AVE.    SYL.            READ
            SENT-             SYL-        SENT    PER    FLESCH   GRADE
FILENAME    ENCES    WORDS    LABLES      LENGTH  WORD   INDEX    LEVEL
------------------------------------------------------------------------

NO21            6     121      221        20.2    1.8     31.8    15.7
NO22            7     116      226        16.6    1.9     25.2    16.7
NO23            7     115      200        16.4    1.7     43.0    14.0
NO24            8     104      218        13.0    2.1     16.3    18.1
NO25            6     120      221        20.0    1.8     30.7    15.9
TOTAL          34     576     1086        16.9    1.9     30.1    16.0
------------------------------------------------------------------------
```

In general, STAR does the following:

1. The number of sentences is determined by counting end of sentence characters. These include periods, exclamation marks, question marks, colons, or semi-colons.
2. Words are identified as fields (one or more characters) separated by spaces and are accumulated.
3. The program counts syllables by counting non-consecutive vowels (A, E, I. O, U, Y), subject to the following exceptions:

   A final "E" is not counted except when preceded by an "L". If the "L" is preceded by a vowel, the final "E" is not counted.

   A final "ED" is not counted except when preceded by an "E", "T", "D", or "L". An "LED" is not counted if preceded by a vowel.

   A final "ES" is not counted except when preceded by an "E", "S", "Z", or "L". An "LES" is not counted if preceded by a vowel.
4. The Flesch Reading Ease Index (R.E.) is calculated.

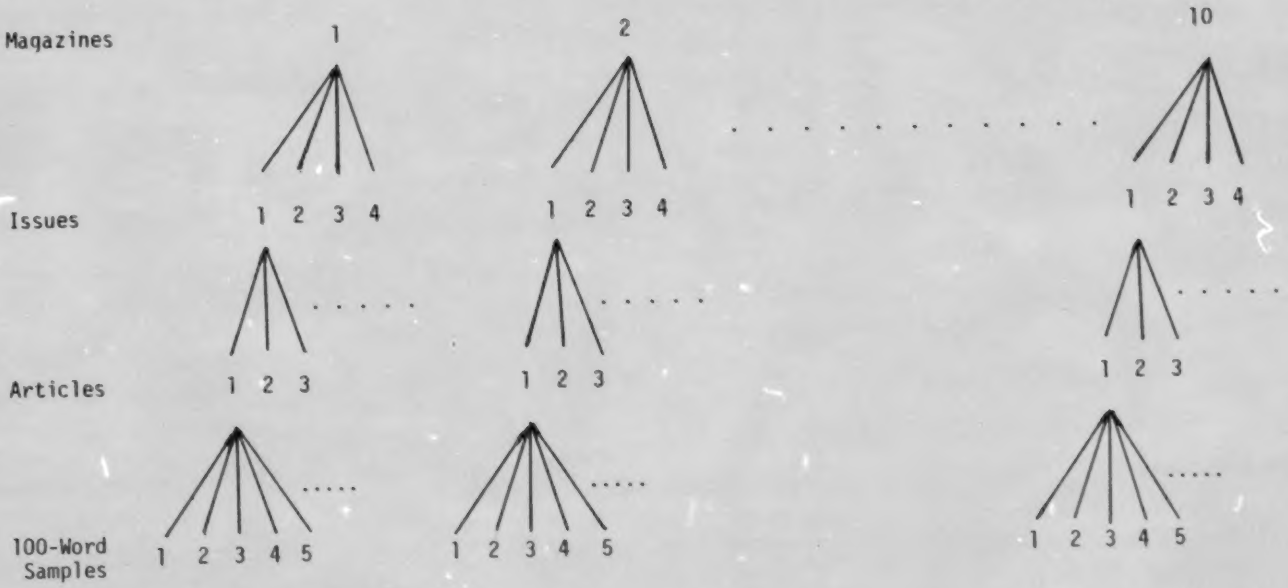
5. The Reading Grade Level is calculated as a derived Dale Index (Dale-Chall score) based on the computed Flesch R.E. score. The Dale-Chall approach, appearing in 1948, is the second most frequently used formuala (7). It uses a word factor (percentage of words outside a 300 word list) and sentence factor (average sentence length in words).

3.3 Procedure

Table 5 diagrams the sampling population used in this study. As shown in Table 5, a total of 600 (10 × 4 × 2 × 5) samples were analyzed (60 samples from each magazine). For each of the ten magazines, four issues were sampled. These issues were published between July, 1979 and September, 1980. The specific issues are shown in Table 6.

Table 5

Hierarchical Research Design
(10 x 4 x 3 x 5)*



*Note: $S_{i,j,k,l}$ = Flesch score for a particular sample. For example, $S_{7,4,2,3}$ = the Flesch score for Magazine 7, Issue 4, Article 2, Sample 3.

Table 6

Magazine Issues
Sampled

Magazine	Month	Year	Publication Frequency
Infosystems	Jul	79	Monthly
	Oct	79	
	Jun	80	
	Apr	80	
Datamation	Jul	79	Monthly
	Oct	79	
	Apr	80	
	Oct	80	
Computer Decisions	Nov	79	Quarterly
	Jan	80	
	Apr	80	
	Jul	80	
Mini-Micro Systems	Oct	79	Monthly
	Jun	80	
	Apr	80	
	Aug	80	
Computerworld	Oct	79	Weekly
	Jan	80	
	Apr	80	
	Jul	80	
IBM Systems Journal	Jan	79	Quarterly
	Oct	79	
	Jan	80	
	Apr	80	
Communications of the ACM	Jan	80	Monthly
	Mar	80	
	Jul	80	
	Sep	80	
Data Management	Nov	79	Monthly
	Jan	80	
	Mar	80	
	Jul	80	
Computer	Oct	79	Monthly
	Jan	80	
	Mar	80	
	Jul	80	
Computer Journal	Aug	79	Monthly
	Nov	79	
	May	80	
	Aug	80	

4. Results

Table 7 shows the overall descriptive statistics relating to the text processing by STAR. Histogram inspection showed the Flesch Index (R.E.) scores to be normally distributed.

Table 7

Descriptive Statistics
For the 10 Magazines

Magazine	Flesch Mean	sd	Range	95% Confidence interval for mean
IBM Systems Journal	26.88	14.71	-11.0 - 53.6	23.08 - 30.68
Computer	28.42	11.03	-15.4 - 47.9	25.57 - 31.27
Computerworld	30.24	17.25	-18.1 - 70.5	25.78 - 34.69
Data Management	30.55	13.66	-11.5 - 59.0	27.02 - 34.08
Infosystems	33.09	14.57	1.0 - 70.4	29.32 - 36.85
Datamation	33.44	14.12	3.4 - 62.7	29.79 - 37.09
Mini-Micro Systems	34.43	15.38	-3.3 - 66.3	30.46 - 38.41
Computer Decisions	37.31	14.69	3.7 - 68.6	33.51 - 41.10
Computer Journal	39.55	12.42	7.1 - 72.9	36.34 - 42.76
Communications of the ACM	43.19	15.30	-6.5 - 80.4	39.24 - 47.14
Combined	33.71	15.09	-18.1 - 80.4	

Table 8 breaks down the mean for each magazine into issue and article means. These data are used in the following discussion of the research design used in this study to indicate the subsampling variance components.

Table 8

Table of Means

IBM Systems Journal
M= 26.88

Issue		Article Means		
No.	Mean	1	2	3
1	19.7	15.9	9.6	33.6
2	27.3	37.4	21.2	23.1
3	26.2	31.5	14.0	33.1
4	34.4	41.6	29.4	32.1

Computer
M= 28.42

Issue		Article Means		
No.	Mean	1	2	3
1	31.4	29.0	36.9	28.4
2	29.7	29.4	22.9	36.8
3	25.7	23.5	25.3	28.3
4	26.8	24.1	30.1	26.3

Computerworld
M= 30.24

Issue		Article Means		
No.	Mean	1	2	3
1	24.7	15.5	6.9	51.6
2	31.8	16.8	32.5	46.2
3	26.3	30.1	22.0	26.8
4	38.1	36.3	39.3	38.8

Data Management
M= 30.55

Issue		Article Means		
No.	Mean	1	2	3
1	29.8	29.0	29.4	31.0
2	27.1	39.2	10.7	31.4
3	27.6	29.1	23.1	30.7
4	37.7	36.2	45.0	31.8

Infosystems
M= 33.09

Issue		Article Means		
No.	Mean	1	2	3
1	29.8	32.3	36.4	20.8
2	35.2	36.0	38.2	31.4
3	27.5	17.0	30.8	34.9
4	39.8	59.7	28.5	31.2

Datamation
M= 33.44

Issue		Article Means		
No.	Mean	1	2	3
1	33.5	48.2	27.7	24.6
2	39.2	42.1	34.5	40.9
3	24.2	20.3	32.9	19.2
4	36.9	40.7	38.9	31.2

Mini-Micro Systems
M= 34.43

Issue		Article Means		
No.	Mean	1	2	3
1	39.9	19.7	41.4	58.6
2	37.1	45.5	36.4	29.4
3	30.4	33.2	19.4	38.8
4	30.3	35.6	29.3	26.0

Computer Decisions
M= 37.31

Issue		Article Means		
No.	Mean	1	2	3
1	34.0	36.3	36.6	29.0
2	42.1	51.2	37.6	37.4
3	37.1	27.5	43.0	40.9
4	36.1	12.7	45.6	50.0

Computer Journal
M= 39.55

Issue		Article Means		
No.	Mean	1	2	3
1	39.7	38.2	38.7	42.1
2	37.2	36.0	39.4	36.4
3	38.7	43.3	47.9	25.0
4	42.5	38.6	54.8	34.1

Communications of the ACM
M= 43.19

Issue		Article Means		
No.	Mean	1	2	3
1	44.9	52.0	34.9	47.8
2	50.9	44.9	60.5	47.4
3	40.3	39.8	32.4	46.6
4	36.7	39.0	37.2	33.9

ANOVA analysis is shown in Table 9.

Table 9

Nested Anova for
Flesch Scores Summary

Source	SS	df	MS	F	p
Magazines				7.54	.000*
Between	14072.7	9	1563.6		
Within	122400.8	590	207.5		
Within Magazines					
Among Issues				1.94	.002*
Between	11513.0	30	383.8		
Within	110887.0	560	198.0		
Within Issues					
Among Articles				3.27	.000*
Between	39140.2	80	489.3		
Within	71747.4	480	149.5		

*p <.05

With a statistically significant F test for the Flesch mean between the ten magazines, post hoc comparisons were used to determine the specific significant differences. These techniques are described in Kirk (6). Table 10 presents the differences between magazines used in this analysis.

Table 10

Differences Between
Flesch Means

	IBM Systems Journal	Computer	Computer-world	Data Management	Info-Systems	Datamation	Mini-Micro Systems	Computer Decisions	Computer Journal	Communications of the ACM
IBM Systems Journal	-	1.54	3.36	3.67	6.21	6.56	7.55	10.43	12.67	16.31
Computer		-	1.82	2.13	4.67	5.02	6.01	8.89	11.13	14.77
Computerworld			-	.31	1.85	3.20	4.19	7.07	9.31	12.95
Data Management				-	2.54	2.79	3.88	6.76	9.00	12.64
Infosystems					-	.35	1.34	4.22	6.46	10.10
Datamation						-	.99	3.87	6.11	9.75
Mini-Micro Systems							-	2.88	5.12	8.75
Computer Decisions								-	2.44	5.88
Computer Journal									-	3.64
Communications of the ACM										-

These data may also be presented as homogeneous subsets whose highest and lowest means do not differ by more than the shortest significant range. For example, the HSD test partitions the means into the following four subsets of statistically similar means.

Subset 1:

- IBM Systems Journal
- Computer
- Computerword
- Data Management
- Infosystems
- Datamation
- Mini-Micro Systems

Subset 2:

- Computerworld
- Data Management
- Infosystems
- Datamation
- Mini-Micro Systems
- Computer Decisions

Subset 3:

- Infosystems
- Datamation
- Mini-Micro Systems
- Computer Decisions
- Computer Journal

Subset 4:

- Computer Decisions
- Computer Journal
- Communications of the ACM

Discussion

First of all, it is important to acknowledge the limitations of the so-called readability indexes. It should be reemphasized that such indexes are only predictors of reading difficulty. These formulas do not account for other important factors in readability such as interest level of the material, syntax of the passage, motivation of the reader, concept load of the vocabulary, and use (or lack of use) of illustrations and examples. Also, different formulas have been found to give different results.

However, if these limitations are kept in mind, readability indexes can be useful tools in analyzing text materials, especially in identifying ordinal relationships. Also, according to Klare (7, P. 97), the two simple variables of word length and sentence length give good predictions of readability.

The IBM Systems Journal (a technical journal) had the lowest readability index (R.E. = 26.88) and Communications of the ACM (also a technical journal) had the highest (R.E. = 43.19). Surprisingly, of the four technical periodicals included in this study, two were found to have had the lowest readability indexes (IBM Systems Journal and Computer) while the other two (Communications of the ACM and Computer Journal) were found to to have the highest readability indexes.

Table 3 showed the computer analysis of a "typical" passage in terms of readability estimates for all magazines (R.E. = 31.8 versus grand mean for all magazines equal to 33.7). Tables 12 and 13 give examples of text passages representing "extreme" scores. Table 12, the first example (from Communications of the ACM), shows the highest readability estimate in this study.

Table 12

Example of Text Sample
With High Readability*

```
-----------------------------------------------------------------------
NAME OF SAMPLE TEXT          M7             80/11/06.
-----------------------------------------------------------------------
TEXT

CHOOSE ANY LINK IN THE CHAIN CODE AT RANDOM AND CREATE A NODE FOR IT, SAY
PEE. NOW, EXAMINE THE NEXT LINK IN THE CHAIN CODE, SAY NEW, AND COMPARE ITS
 DIRECTION WITH THAT OF THE IMMEDIATELY PRECEDING LINK, SAY OLD. THERE ARE
THREE POSSIBILITIES. ONE, IF THE CODE MAINTAINS ITS DIRECTION, THEN WE MAY
NEED TO ADD A NODE, SAY QUE, WHICH IS A NEIGHBOR OF PEE IN THE OLD
DIRECTION, SEE FIGURE TWO-A. TWO, IF THE CODE TURNS CLOCKWISE, THEN THERE IS
 NO NEED TO ADD A NODE, SEE FIGURE TWO-BEE. THREE, IF THE CODE TURNS
COUNTERCLOCKWISE, THEN WE MAY HAVE TO ADD TWO NODES.
-----------------------------------------------------------------------
HI-CAL WORDS

 1 COUNTERCLOCKWISE           3 DIRECTION    1 EXAMINE      1 IMMEDIATELY
 1 POSSIBILITIES              1 PRECEDING
-----------------------------------------------------------------------
SUMMARY AND CALCULATIONS

NUMBER OF SENTENCES=   6.0  AVERAGE SENTENCE LENGTH=  17.8
    NUMBER OF WORDS= 107.0  AVE. SYLLABLES PER WORD=   1.3
NUMBER OF SYLLABLES= 137.0              FLESCH INDEX=  80.4
            READING GRADE LEVEL=   7.0
-----------------------------------------------------------------------
```

*Source: Samet, H. Region representation: quadtrees from boundary codes. Communications of the ACM, March 1980, p. 165.

As shown, high readability is attributable to relatively few words with 3-or-more syllables and short sentence length. The second example, Table 13 (from Computer), shows a low readability extreme.

Table 13

Example of Text Sample
With Low Readability*

```
-------------------------------------------------------------------------------
NAME OF SAMPLE TEXT          CSSE1          80/12/11.
-------------------------------------------------------------------------------
TEXT

HOWEVER, A COMBINATION OF MANY FACTORS, INCLUDING THE TREMENDOUS GROWTH IN
THE NUMBER OF STUDENTS TAKING COMPUTER SCIENCE COURSES, THE INCREASE IN
TWO-YEAR INSTITUTIONS AND SMALLER COLLEGES AND UNIVERSITIES INITIATING
COMPUTER SCIENCE PROGRAMS, AND THE RAPID CHANGE IN COURSE CONTENT AND THE
INTRODUCTION OF NEW COURSES, HAS RESULTED IN A NUMBER OF SUBSEQUENT
PROPOSALS FOR COMPUTER EDUCATION CURRICULA. WITHIN THE AREA OF COMPUTER
SCIENCE THERE HAVE BEEN SEVERAL RECENT MAJOR CURRICULUM EFFORTS INCLUDING
AN UPDATED SURVEY, A REVISION OF CURRICULUM SIXTY-EIGHT, A DRAFT OF A
CURRICULUM DESIGNED FOR TWO-YEAR INSTITUTIONS, AS WELL AS A PROPOSAL
DEVELOPED UNDER THE AEGIS OF THE I-EE-EE-EE COMPUTER SOCIETY EDUCATION
COMMITTEE.

-------------------------------------------------------------------------------
HI-CAL WORDS

1 COMBINATION  1 COMMITTEE     5 COMPUTER     1 CURRICULA    3 CURRICULUM
1 DEVELOPED    2 EDUCATION     1 HOWEVER      2 INCLUDING    1 INITIATING
2 INSTITUTIONS 1 INTRODUCTION  1 PROPOSAL     1 PROPOSALS    1 RESULTED
1 REVISION     1 SEVERAL       1 SIXTY-EIGHT  1 SOCIETY      1 SUBSEQUENT
1 TREMENDOUS   1 UNIVERSITIES  1 UPDATED
-------------------------------------------------------------------------------

SUMMARY AND CALCULATIONS

NUMBER OF SENTENCES=   2.0   AVERAGE SENTENCE LENGTH=  53.0
   NUMBER OF WORDS= 106.0   AVE. SYLLABLES PER WORD=   2.0
NUMBER OF SYLLABLES= 211.0              FLESCH INDEX= -15.4
              READING GRADE LEVEL=  22.8
-------------------------------------------------------------------------------
```

*Source: Aiken, R. Computer Science in the 1980's. Computer, Jan 1980, p. 41.

Notice how there are many multiple-syllable words and that the passage consists of two long sentences. Readability theory suggests that even a person who is not content-familiar with the material in these paragraphs, could perceive the significant differences in readability levels.

As mentioned earlier, the Flesch Index can be transformed into estimates of reading grade level equivalents. These data are shown in Table 14.

Table 14

Reading Grade Levels
of Computing Periodicals

Periodical	Flesch R.E.	Reading Grade Level	SD	Range
IBM Systems Journal	26.88	16.46	2.23	11.9 - 22.2
Computer	28.42	16.24	1.66	13.3 - 22.8
Computerworld	30.24	15.83	2.86	8.0 - 23.2
Data Management	30.55	15.87	2.16	10.3 - 22.2
Infosystems	33.09	15.41	2.47	8.0 - 20.4
Datamation	33.44	15.37	2.35	9.5 - 20.0
Mini-Micro Systems	34.43	15.18	2.61	8.7 - 21.0
Computer Decisions	37.31	14.75	2.46	8.3 - 19.9
Computer Journal	39.55	14.38	2.26	7.7 - 19.4
Communications of the ACM	43.19	13.74	2.72	6.96- 21.48
Combined	33.71	15.32	2.51	6.96- 23.22

In general, a combined reading grade level estimate of 15.32 suggests that a person needs about 3-1/3 years of college to potentially deal with the major computing periodicals in an effective manner. The range of means indicates the need for almost two years of college (Grade = 13.74 for Communications of the ACM) to almost 4-1/2 years (masters-level) for IBM Systems Journal (Grade = 16.46).

The reader is left to draw his/her own conclusions as to the appropriateness of the apparent degree of difficulty in readability evident in the magazines analyzed.

Readability indexes like R.E. can indicate potential barriers to effective communciation of ideas. However, just as eliminating GOTOs does not assure a better program, shortening sentence length and using words with fewer syllables will not ensure improved readability.

Another positive outcome of using readability indexes is that a sampling scheme can highlight problems of variability in text readability. For example, Computerworld showed the greatest overall variability in readability indexes. The standard deviation was the highest at 17.25. This was accompanied by a R.E. range of -18.1 to 70.5 (Grade Level: 23.2 to 7.95). In contrast, Computer showed significantly less variation with a standard deviation of 11.03 and R.E. range of -15.4 to 47.9 (Grade Level: 22.8 to 13.3). Consistency of writing style within articles, within issues, and between issues can be important measures of readability.

In conclusion, it is proposed that readability indexes can be of practical importance for both readers and publishers of computing periodicals.

References

1. Dixon, W. and Brown, M. (editiors). BMDP-79 Biomedical *Computer Programs P-series*. University of California Press, Berkeley, CA, 1979.

2. Greenberg, H. (editor). *The Standard Periodical Directory*, Sixth Edition, Oxbridge Communications, Inc., NY, 1979-80.

3. Flesch, R. *How to Test Readability*. New York, Harper, 1951.

4. Flesch, R. A new readability yardstick. *Journal of Applied Psychology*, June 1948, 221-233.

5. General Motors Corp. S.T.A.R. brochure. General Publicity, 3044 W. Grand Boulevard, Detroit, Michigan 48202.

6. Kirk, R. *Experimental Design: Procedures for the Behavioral Sciences*. Brooks/Cole Publishing Company, Belmont, CA, 1968

7. Klare, G. Assessing readability. *Reading Research Quarterly*, Number 1, 1974-75, 62-102.

8. Lemos, R. STAR: a computerized readability analysis tool. Working Paper, California State University, Los Angeles, 1981.

9. Lemos, R. CDC CYBER BASIC version of STAR. Available from the author, 1980.

10. Levine, H. Robust tests for equality of variances. In I. Olkin, ed., *Contributions to Probability and Statistics*, Palo Alto, Stanford University Press, 1960, 278-92.

11. Nie, N. *SPSS: Statistical Package for the Social Sciences* (second edition). McGraw Hill, New York, 1975.

12. *Ulrich's International Periodicals Directory*, 18th edition. R.R. Bowker Co., 183 Madison Avenue, NY, 1979-80.

An Interactive, Instructional, CODASYL Data Base Management System For Main Frames and Microcomputers

by
Dr. Marino J. Niccolai
University of South Alabama
H. Rylan Lewis
University of Louisville

1.0 INTRODUCTION

This paper describes the design concepts and possible uses of an instructional Data Base Management System (DBMS). This system was designed and implemented at the University of Louisville. It was developed under the auspices of the department of Applied Mathematics and Computer Science (AMACS) of the Speed Scientific School and is titled the AMACS Data Base Management System (ADMS). The ADMS is an on-line system and follows the CODASYL DBTG data base model. It was initially implemented on the university's DECsystem-10 mainframe computer using Pascal and was ultimately downloaded to an Apple II microcomputer. The primary purpose of the ADMS is to provide "han on", laboratory experience to data base students as well as to support student and faculty experimentation. Major design considerations were a high degree of modularity and open-endedness to facilitate future changes and portability. In the process of surveying potential data structures, an Instructional Data Base Management System (IDBS) was obtained from Colorado State University. The IDBS is a CODASYL DBMS and was developed by Dr. David Kroenke who is the author of several articles and texbooks on data bases. The ADMS borrows several concepts and data structures from the IDBS [1], but ADMS is essentially a simpler design intended to allow for substantial student/faculty interaction and upgrading.

The CODASYL proposal describes a very complicated data base model and has been the subject of entire books. The proposal is well written and describes the model in great detail but does not address questions of implementation and potential problems. Because of the complex nature of the CODASYL model, the intention of this project was not to implement the entire DBTG proposal by any means, but only the necessary fundamentals. The ADMS is instructionally useful without being an extremely sophisticated DBMS. The design was simplified not only due to development time considerations but also due to the desirability of simplicity for classroom demonstrations and future changes to the code itself. For this latter reason, the ADMS was developed in a highly modular fashion using top-down refinement techniques. Rather than developing a system with a few sophisticated features, such as a hashed key retrieval, the ADMS was developed in a manner which incorporated many less complicated, straightforward features which were more fundamental to the subject of data base itself. The ADMS supports such advanced features as variable length records and partitioned data sets. It does not support variable length or ordered sets, and it does not have any Key Retrieval techniques as such, but there is a special feature (LIST) which can be used as a crude, non-unique key retrieval method.

2.0 PORTABILITY AND MICROCOMPUTER DBMS DESIGN

The ADMS was designed in a highly modular fashion to facilitate ease of portability. It was also parameterized to the extent that a system programmer should be able to go to a few well defined program locations and make the necessary changes for the intended host computer. The ADMS design makes the following assumptions for the host system:

(1) The system supports a direct access to disk on a block level.
(2) The physical block size is no smaller than 256 bytes.
(3) The system CRT terminal is at least a 24 x 80 character display.

(4) The system supports the ASCII code.
(5) File names, with extensions, are no longer than ten characters.

To transport the ADMS from one computer to another requires that only four declared constants and five short subroutines be changed. Two other short subroutines should be suspect for possible changes since they use intrinsinc Pascal functions which are compiler dependent. A test of this portability was made by loading the ADMS onto an Apple II microcomputer. After changing the six prime elements, it was found that there was one major problem with the two compilers: the Apple Pascal used a subset of the DECsystem-10 Pascal. The PACK and UNPACK commands were not included on the Apple system. Once the problem was diagnosed, subroutines were written to emulate the missing instructions.

2.1 DATA BASE ON A MICROCOMPUTER

The microcomputer constitutes a unique and challenging problem for the world of DBMS. These DBMS programs are notoriously very large, complex, and require large allocations of computer resources. This is precisely what is not available on the microcomputer. A software engineer faces limited core, less powerful Operating Systems, and less secondary storage when designing for microcomputers. There are currently only a few psuedo-microcomputer DBMS available on the market including Relational and CODASYL models.

During the course of the project a literature search revealed several articles on the subject of microcomputer data bases; however, the predominance of these articles addressed file managers rather than true data bases, even though data base was implied in the title. These systems were mainly concerned with sorting, report generation, and key retrieval techniques; they did not address any relationships between files.

One reason that true DBMS are rare on microcomputers is that the smaller systems use floppy disk secondary storage. This is a very popular hardware configuration for a small business or a home user. The floppy disks have limited storage capacity compared to hard disks. As file lengths become smaller, sequential searches over entire files are as effective in finding records as using data base organized relationships. As file sizes grow the data base approach becomes more advantageous. Since floppy disk files are so small, a data base system using them is not useful unless the files and relationships can be implemented in a cross-volume design which would then allow large files.

One goal of this project was to implement the ADMS on a microcomputer and examine the microcomputer data base problem. The ADMS was downloaded to the Apple II and the conversion was made as previously described. However, this system can only utilize a single volume (diskette). Therefore, slight modifications were proposed to allow a multivolume data base. The proposed design was successful and it will be briefly described. The basic idea is to let each file reside on a single diskette as shown in Figure 1. This required minimal change to the ADMS design, consisting entirely of inserting pauses to allow switching the diskettes. Of course the main problem to be expected is the inconvenience of having to do this many times. The success of this approach proves that true DBMS can work on single drive floppy disk microcomputer systems. However, these files are still very small and may not even prove to be competitive with sequential searches. The files must be made larger to claim that data base is viable on small floppy disk systems. The proposed solution to this problem is to allow each data file to grow to multiple volumes. This next level of expansion may not be tenable, but a conjectured solution would require an added level of indexing. This new index would be stored along with any pointers and would indicate on which volume the desired record resides. If this design is viable the file size can then grow large enough to insure that the data base would out perform a sequential search file manager. The next higher level of power in microcomputer data bases would certainly require a hard disk.

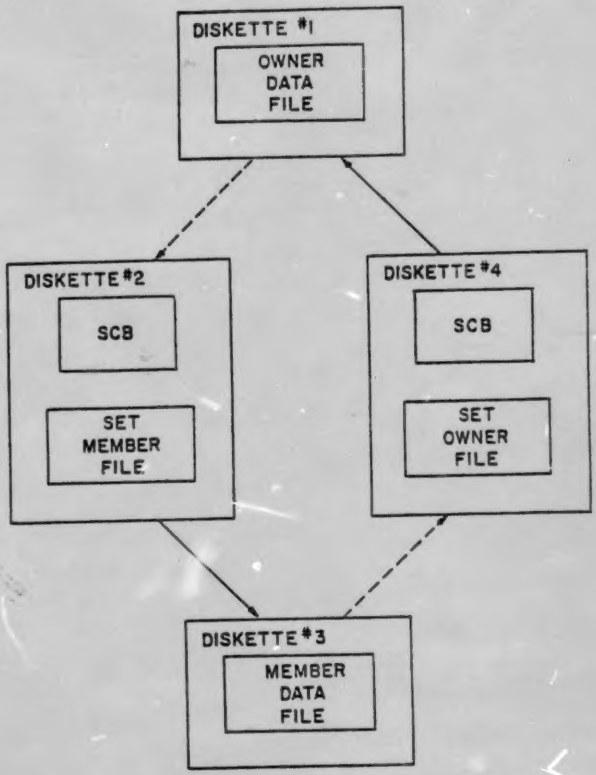


FIGURE I - Multivolume Floppy Disk Database

3.0 SYSTEM DEFINITIONS

Throughout this paper several terms will be used extensively and will be defined at this time. These definitions are taken directly from the CODASYL DBTG proposal [2]. A DATA-ITEM is the smallest unit of data possible. It corresponds to what is conventionally called a field. A RECORD is a collection of data items. It is a logical concept and has no correspondence to the physical block of a file - a FILE is a collection of (logical) records. A RECORD TYPE is the logical collection or structure of a group of records, and a RECORD OCCURRENCE is the data (record) conforming to that structure. In general, files are a collection of Record Types.

A SET is a collection of occurrences of records of a particular Type or Types. A Set defines a one-to-many relationship between different Record Types. A set has an OWNER which is a record occurrence of one Record Type, and a MEMBER which is an occurrence of another Record Type. This allows each record occurrence of the owner Record Type to be associated with one or more member occurrences of a member Record Type.

An example of a Set is given in Figure 2. The relations for a particular Department record are a SET OCCURRENCE, and the total collection of relations constitute a SET TYPE. The double arrowhead represents a one-to-many record occurrence relationship. Several record occurrences of Record Type Faculty are related to a single record occurrence in Record Type Department. This is an "Owner-Member" relationship. A record in Record Type Faculty cannot be owned by any other record occurrence of Record type Department which is in the same Set Type. More than one Set Type can exist between two Record Types. For example, a Department/Chairman relation can also exist for the Record Types shown.

The ADMS does not adhere stringently to the CODASYL proposal, but uses it as a guide rather than a specification. One major difference is that the DBTG allows files of non-homogeneous Record Types. The ADMS will allow this configuration but not in an automatic mode; it must be done on an application program level. ADMS data files are collections of homogeneous Record Types (i.e a classical file manager data file). The ADMS Set Types can only have members from one Record Type while the DBTG allows Sets with member records from several Record Types. As with the DBTG model, a record can own any number of Set Occurrences of different types, but it can only own one occurrence of a given Set Type. In DBTG terms, all records are OPTIONAL TRANSIENT set members. This means that ADMS does not automatically insert or delete any records into or from any sets. The user application program must do this.

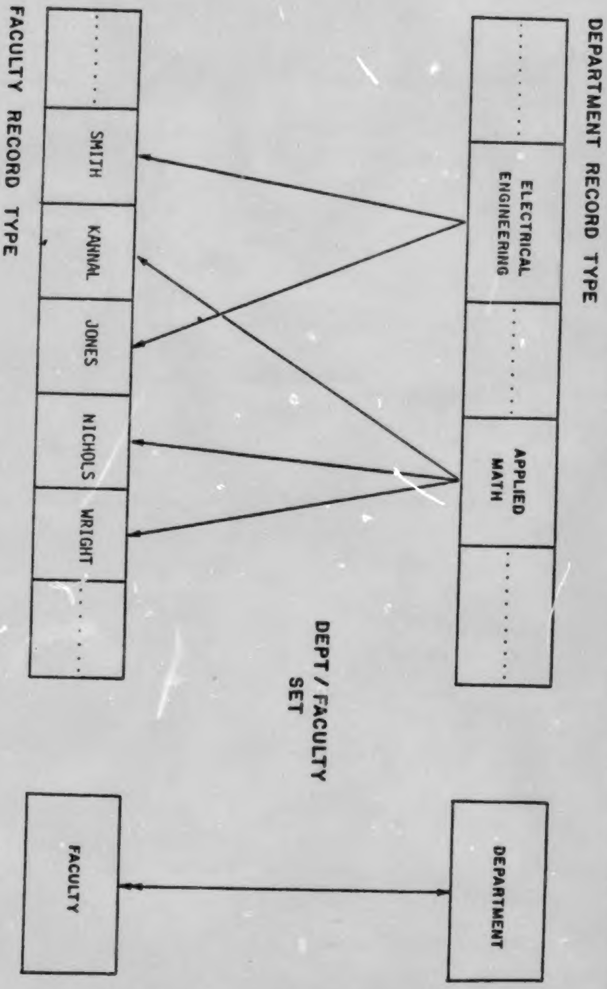
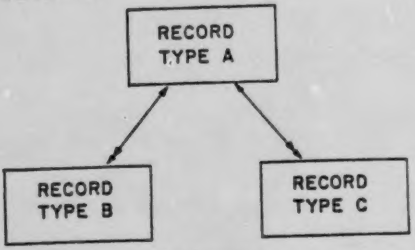


FIGURE 2 - Set Example and Representation

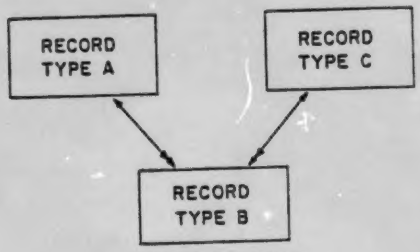
3.1 DESIGN OVERVIEW

There are currently three main types of data bases: Hierarchical, CODASYL, and Relational. Most commercial data bases are some combination of these three types. Data bases differ from the concept of file managers in that data bases allow direct relations between files using an instruction set. The theoretical description of the data base is the SCHEMA which describes the Record Types and/or relations that comprise the data base. The term data base usually refers to an occurrence of a schema, and several schemas are referred to in the plural--data bases. A data base management system should operate on an arbitrary collection of schemas.

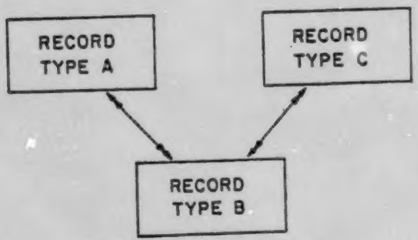
All of the three types of data base models support sets; however, only the CODASYL and Relational models support the network configuration of the sets. This is shown in Figure 3. In the Hierarchical model, record ocurrences of Record Type B can only be owned by records from Record Type A. In the network model, records from Record Type B can belong to sets owned by more than one Record Type, such as Record Type C. The CODASYL model uses physical pointers to implement this configuration while the Relational model does so by ordering the data in storage in a particular fashion. In the complex network, records from Record Type B can have more than one owner record within a Set Type, as shown by the additional double arrowheads. The complex network is only allowed in the Relational model. This paper is concerned only with the CODASYL model simple network configuration.



HIERARCHICAL



SIMPLE NETWORK



COMPLEX NETWORK

FIGURE 3 - Database Organizations

3.2 DESIGN IMPLEMENTATION

The DBTG proposal provides three alternative means for accessing records: currency status, logical position, and location mode. The ADMS supports all three access methods to some extent. The concept of CURRENT STATUS is paramount in understanding the ADMS mechanics. Records can be identified as the FIRST, NEXT, PRIOR, or OWNER record of either CURRENT-OF-FILE or CURRENT-OF-SET. By using CURRENCY INDICATORS it is possible for a user to "traverse" the data base using a Query Program.

Records can also be retrieved using the non-CODASYL command LIST. The LIST function was designed specifically to enhance the instructional aspects of the ADMS. It allows rapid examination of files and set occurrences by displaying data from several records on a single display. It allows the user to view portions (the first 32 characters) of five contiguous records at one time, as if looking at the file or set through a "window". This window can be "scrolled" up and down the file or set and thereby allows the user to examine the file or set without having to retrieve every record. Individual records can be selected for retrieval using the LIST function by inputing the field number of the desired record. The retrieval and scrolling features combine to provide a possible extended feature: if the first 32 characters of each record are reserved for some type of discriminatory information, then the LIST could roughly serve as a non-unique key retrieval system.

The DBTG suggests two set implementation designs: chained members and pointer array constructs. It was decided that a pointer array design would be a more straightforward approach and would eliminate many of the complex program mechanics. This procedure would allocate two data base files to hold each Set Type pointer array. The overall concept is shown graphically in Figure 4 and again in more detail in Figure 5. The broken arrow means that record relationships are mathematical; physical pointers are not used. The Set Control Block is used to store the set parameters. The other two set files contain the pointer arrays - one for each set occurrence. As Set Types are added and deleted from the schema, these three files would be added or deleted from the data base.

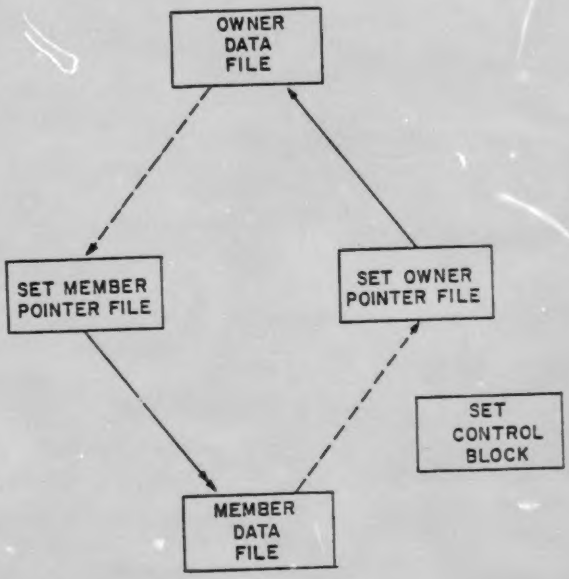


FIGURE 4 - ADMS Set Implementation Overview

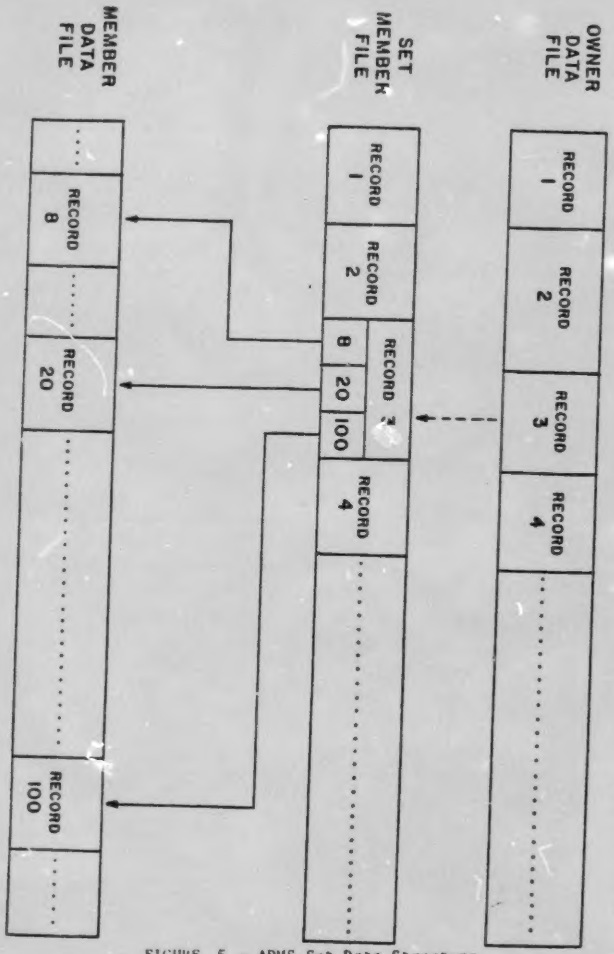


FIGURE 5 - ADMS Set Data Structure

If the set members for the occurrence owned by record number three of the Owner File are to be retrieved, the ADMS first retrieves record number three of the Set Member File. Therein it finds the array of pointers to the appropriate Member Data File records. The same process is repeated for the Owner relation which has its pointer array stored in its own file. Each array could be searched sequentially, and its size could be allocated upon creation.

3.3 PROPOSED MODIFICATIONS FOR CLASSROOM DEMONSTRATIONS

The actual implementation is somewhat more complicated than this due to the Partitioned Data Sets, but this is a good overview. The current version of the ADMS does not support ordered sets; however, should this feature ever be desired, the pointer arrays could be treated as a push-down stack to insert a new pointer where necessary. This method could also easily support a key retrieval system such as an indexed-sequential or hashing scheme.

There is an additional constraint on the Owner relationship in that the Owner pointer array can only have one entry by virtue of the CODASYL set definition: a member record of a Set Type can have only one owner. Since the ADMS does not have variable length sets, this situation causes an entire array to be allocated to contain a single pointer. This is the worst flaw in the system.

The ADMS is composed of three major program elements: a Query program (QUERY.PAS), a Utility Program (DBUTIL.PAS), and a Data Base Manager Subroutine (DMS). QUERY.PAS and DBUTIL.PAS are stand-alone programs while DMS is an internal subroutine imbedded in QUERY.PAS. THe Utility Program provides functions to initialize a data base, allocate and delete files and sets, and a diagnostic dump routine which allows visual inspection of part of the data base. The Query Program has functions which perform the basic instructions of the CODASYL DBTG proposal. The Data Base Manager Subroutine is the system element which actually implements the CODASYL Data Manipulation Language. In the ADMS, it is driven by QUERY.PAS, but it can easily be used in an application program.

The Query Program uses three Functional Displays to perform the CODASYL commands. The displays are organized as shown in Figure 6. The arrows indicate control pathways that are available to the user. The Main Display is only an entry/exit point to the system. The File functions and Set functions are performed by selecting CODASYL commands from a menu in the respective Functional Display.

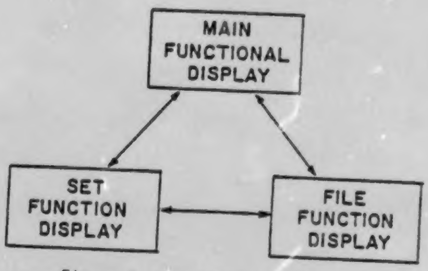


Figure 6 - Functional Displays

4.0 CONCLUSIONS

A useful and portable version of a DBMS could be used as an on-line interactive teaching device for a number of different purposes. First, as an introduction to CODASYL, the system would allow the structure of sets and relationships to be viewed and manipulated in a working environment. Second, application type programs could be easily developed to simulate the environment of "real-world" system. Finally, at a more advanced level, the ADMS could be expanded or modifed to include any desired data base features.

The significant contribution of ADMS in an instructional environment is the ability to download to a microcomputer and still retain the concept of sets and relationships. Much work is required to develop a commercially useful "True" micro-data base management system;

however, this project provides a start and establishes the possibilities for a floppy system. Because of the rapidly increasing population of microcomputer laboratories in educational environments, this model and design could prove fertile ground for additional work on data base systems.

REFERENCES:

1. Korenke, D., "IDBS Instructional Data Base System, Version 1.1," June 1977, A users manual to a computer program purchases from Colorado State University, Fort Collins, Colorado.

2. "CODASYL Date Base Task Group, April 1971 Report, ACM, New York, 1971.

USING ROBOTS TO HELP TEACH ASSEMBLY LANGUAGE

Jerry Scott
Department of
Computer Science
University of South
Alabama Mobile, Al.
36688 1-205-460-6390

1.0 INTRODUCTION PROBLEMS IN LEARNING ASSEMBLY LANGUAGE

One of the problems in teaching assembly language is the amount of background material a student must learn before even a simple program can be understood. This problem is crucial on mainframe assemblers like the IBM 370 OS assembler. One solution is to develop small subprograms called "Macros" to handle the difficult transition areas like input/output. These macros are then incorporated into the assembler and can be used as needed. An IBM 370 assembler of this type is Assist, which was developed at Penn State in the early seventies. Assist has useful macros to handle input/output and register dumps. In addition, Assist runs much faster than the standard OS assembler.

Others suggest that microcomputers should be used to begin learning assembly concepts. They note that microcomputers are architecturally simpler than mainframes and so are their assembly languages. Most students come to a first assembly course with a fair programming knowledge of at least one language, usually Basic or Fortran.

Learning assembly is probably more important to the potential microcomputer programmer than it is for the mainframe programmer. Three reasons for this are:

1) Since microcomputers are much slower than mainframes, speedup procedures are often necessary. When this happens, assembly is dictated. This is particularly the case in Arcade Games programming, where interpreted Basic is too slow.

2) Memory is usually not a major factor on mainframes. Compilers often optimize codes on mainframes. Thus a mainframe programmer can confidently program in a high level language like PLI knowing that the object code produced by the compiler will be nearly as good as what he could produce from assembly language.

3) Some compilers exist for microcomputers but their speed gains come at the expense of more memory used. The Applesoft compilers for the Apple II usually produce object programs which run 2 to 3 times faster. But these object programs typically comsume 2 to 6 times more precious memory. This means that long programs can't be compiled. Usually these same programs written entirely in assembly will run 2 to 5 times faster than the same Applesoft program compiled.

To bring a student from being able to program in Fortran or Basic to being able to effectively use a macro-less microcomputer assembler requires a good deal of effort both from the teacher and the student. This process is difficult even on a well documented system like the Apple II. There are several user friendly assemblers and an excellent system manual. Both the 6502 processor architecture and Apple II Monitor Rom are completely documented and annotated. There are also separate books describing in great detail the Disk Operating System and the Applesoft Roms. An Atlas of all 64K memory locations and their system use has also been provided. A list of these very helpful items is in the bibliography.

2.0 ROBOT WARS BACKGROUND INFORMATION

It was with this background that the "Robot War" experiment was carried out in CS310 "Introduction to Microcomputers" during the summer of 1981. The only prerequisite for CS310 is one programming course, usually in Fortran, but most students have had several courses before they take CS310. This course teaches microcomputers and hardware with most programming emphasis in 6502 assembly language.

"Robot Wars" is a program for the Apple II written by Silas Warner, and distributed by Muse Software. The robots in this program only perform on a rectangle within the the Apple II screen. This rectangle has walls and corners. The robots "live" in a futuristic society where people fight with each other only through robots. All robots in this society have exactly the same hardware.

Several class days were spent learning this program, and then a tournament was held at the end of the course. We will describe how the Robot War program works, the students' reaction to using it, and suggestions for using Robot Wars in a class. The final section is the Bibliography which has useful books on the Apple system.

3.0 ROBOT HARDWARE DESCRIPTION

Briefly, the hardware on each robot consists of:

A) 26 general purpose Registers labelled as Registers A through Z,
B) Special groups of sensor type registers known as
  i) the RADAR register, the AIM register, and the SHOOT register,
  ii) the DAMAGE register, and
  iii) the SPEED X and SPEED Y registers, and
C) Then there are two special programming registers known as the INDEX register and the RANDOM register.

On the field of battle, robots move, aim, shoot, emit radar beams to see where other robots and walls are, and inspect their damage registers to find out if they have been hit. A "Fighting Program" is some logical sequence of manipulating these capabilities. The only thing which makes one robot a better fighter is his fighting program. Programming these robots provides a good introduction to fundamentals of assembly language, simulation, and real time programming concepts.

4.0 ROBOT PROGRAMMING LANGUAGE

The program comes with a very complete manual. With some effort, students can read the manual, learn how a robot functions, and how to program it. This reading experience is good for first year programmers, as reading manuals just within their reach and installing applications programs are a regular activity among programmers.

The "Robot War Source Language" is a mixture of Basic and 6502 assembly language. The fundamentals include direct register addressing, indirect addressing, and pointers. Source programs must be entered with an "Editor" which is actually a streamlined version of "Supertext", a popular Apple II word processor also distributed by Muse. The Editor allows direct screen manipulation, sometimes a first for many students.

Once a source program is finished, the robot must be "Assembled." This process simply translates the source code program into what is known as "Robot Object Code". Only assembled robots can enter the Robot Wars. Once a robot has been assembled, several options are available. The robot can be saved to disk, entered into a single combat, entered into a tournament, or be sent to the "Test Bench."

5.0 THE ROBOT TEST BENCH

The Test Bench simulates a real battle and provides the program's designer with an opportunity to verify his program. On the test bench he can:

A) Inflict damage on the robot to see if its damage routine is really working as it should;
B) Check out the movement routine to see if it is fast enough, or if the robot runs into walls thus damaging itself;
C) Monitor all the registers to see how long it takes to do certain things, like move from point A to point B, or determine how many cycles it takes for the robot to shoot;
D) Monitor the flow of the program, thus discovering ways to speed up the robot, etc.

The test bench is new for most beginning students. It provides a good foundation for later work with trace and debug routines, and memory dump analysis. It also acquaints them with the ideas of program simulation and real time programming. Unfortunately, most students were so eager to get their robots into the Robot Wars that they didn't use the test bench at first. After literally "losing their shirts", they used the test bench and improved their programs.

6.0 CLASS USE OF AND REACTIONS TO ROBOT WARS

The students were assigned to programming teams of about 4 students each. Each student in a particular group would normally receive the same grade for the project. The grades were assigned as follows. The instructor constructed a "C" robot and a "B" robot. A group would receive the grade of the best robot it could beat in a Robot War of three rounds, and be eligible to make an "A" only if it could beat the "B" robot. The champion robot would be the only one to earn his creators an "A" for this assignment.

Programming the robots generated tremendous interest among the students. Many said that this programming was the most enjoyable they had done at the University. They felt a vicarious involvement with their robot. When it lost, they'd rush to patch the code to make it better. Often these same students would submit other programs in CS310 which barely did the job, but not their robot programs.

Students could instantly tell from a Robot War how good their programs were. They never saw the "C" or "B" robots until the last day of the class when the tournament was held. They started out by learning to defeat all the demo robots. It seemed someone would make a "Robot Breakthrough" every few days. His robot would then be the champion until someone else discovered a new programming trick. All the students were charting completely unknown territory, for there are no books on "Robot War Tactics". Other good programmers and other instructors had no idea of what Robot War programming was. Ultimately, the whole problem of fundamental fighting strategy evolved. Since the sensor registers can span 360 degrees, which takes a long time and allows lots of damage to be inflicted by opponents, the first idea was to get to a corner. This reduces the scan to only 90 degrees and makes scanning routines 4 times faster. Then someone figured out how to aim and shoot at the corners, which nullified the corner advantage. Then someone else discovered a superior move routine. An on it went.

All fighting programs are limited to a maximum of 256 bytes. The instructions are "tokenized", so programs could have a maximum of 80-125 instructions. The big enemies in Robot War programming are execution speed and program length. Some of the tricks learned in speeding up the robots were storing often used values in the registers, using the index register, and using pointers to registers. These three ideas helped the students understand the usually confusing concepts of indexed addressing and indirect addressing. The unanimous opinion of the students was that the work with the registers helped them learn 6502 assembly language. Course evaluations often said the Robot War experience was the highlight of the course and should definitely be continued.

7.0 SUGGESTIONS FOR USING ROBOT WAR IN A CLASS

I would offer the following advice to anyone who plans to use Robot Wars in a class.

A) Group the students in small groups of 3-5. Give each student only one of the portions of the Robot Project to build for his team. Examples are the actual coding, the group progress, the test bench analysis and the written writeup of the tournament robot's final strategy. This helps insure that everyone contributes to the group and that the robot doesn't become solely one person's creation.

B) On the first class day, go over the entire program operation, running a battle of three rounds with two demo robots. The program is menu-driven and easy to follow. The complete program instructions should be provided that first day. Conclude the first day by keying in the simple robot given in the program documentation.

C) On the second day, build a robot which moves and avoids hitting the walls. He should still fire at 10 degree intervals. On the third day, change the firing routine so that the robot only fires at robots. Also give him a simple damage routine. Use the test bench to check out all the code. After three days, answer very few questions. There is always the test bench. You'll be pleasantly surprised at the varied approaches your students try.

D) At tournament time, each group hands in the written report containing its progress, the commented code for the finished robot, the report showing how the group used the test bench, and the report describing the robot's fighting strategy. Use this report and their robot's performance to grade each student. Be prepared to give out trophies or plaques to each member of the the winning group, the creators of the "King of the Robot Wars".

E) Most teachers will probably find this class experiment trying, since there are no books to study, and so much strategy is left to the imagination. Getting to the nearest corner as quickly as possible would definitely appear to be the best strategy. (Un)fortunately, you always begin a Robot War on a random coordinate of the battle field.

F) Despite this, the first King of the Robot Wars fought with a corner strategy and had a fairly good damage routine. Suggestions for improvement were to give it a more random damage routine, and a slightly faster moving routine.

G) Try this program! It works and is fun, motivating and stimulating to most students. If you do try it, I'd like to hear from you about your results. Who knows, maybe we can have a ROBOT WAR contest by electronic mail.

8.0 BIBLIOGRAPHY -- USEFUL BOOKS ON THE APPLE II COMPUTER SYSTEM

A) i) The Dos Manual
   ii) System Reference Manual
   iii) The Apple Monitor Peeled, all three by Apple Computer, Inc. The Dos and System manuals are supplied when an Apple is purchased. They are both excellent. The Monitor manual provides great detail on how the monitor routines work.

B) i) 6502 Assembly Programming by Lance Levinthal Osbourne-McGraw Hill
   ii) Programming the 6502 by Rodnay Zaks, Sybex Publishing
   Both these books are good 6502 assembly texts, but are not specific for the Apple. They must be supplemented if the course uses Apples for programming problems.

C) Using 6502 Assembly Language by Randy Hyde, Datamost Publishing
   A really good 6502 Apple assembly text. Has lots of material from system manual, so can easily be used as the sole text for the course. Specific for the "Lisa" assembler, also by Randy Hyde and distributed by On-Line Systems.

D) Beneath Apple Dos by Don Worth and Peter Lechner, Quality Software. The definitive reference book on Apple Dos.

E) All About Applesoft by Call Apple published by same. Call Apple, located in Renton, Wash. is the world's largest Apple user group. This book has much detail on Applesoft and machine language interfaces to it. As such it is the definitive work on Applesoft.

F) What's Where In The Apple by William Luebbert published by Micro Magazine. An atlas to all 64K memory locations used by any Apple. The atlas is invaluable in decoding Basic Peeks and Pokes and other assembly programs.

G) S-C Macro Assembler by S-C Software, Dallas, Texas. This and the "Lisa" assembler are the two easiest to use in teaching 6502 assembly on the Apple. Lisa is an interpreter while S-C supports macros and has much better editing features. I prefer the S-C because of its editing features. S-C also publishes a journal devoted to Apple Assembly Language Programming called Apple Assembly Lines.

Mathematics

FORTRAN AND INTRODUCTORY NUMERICAL METHOD A FRESHMEN COURSE

G.J.Borse and C.W.Clump
Lehigh University
Bethlehem, PA 18015
(215) 861-4025

INTRODUCTION

The core of a beginning engineering curriculum traditionally consists of courses in calculus and physics. In the last decade or so a third area, computer programming, has gained overwhelming acceptance as one of the essential skills prerequisite to later technical studies. An enormous variety of courses have been created to satisfy this need. Most of these introductory courses have concentrated on teaching the elements of Fortran programming and have left the bulk of the techniques of problem-solving to later specialty courses in the various engineering disciplines. The main emphasis of all of these later courses is the teaching of numerical methods, as they apply to each subfield of engineering or science. The numerical methods in these courses are essentially identical, only the example problems differ.

Over the last ten years, the College of Engineering and Physical Sciences at Lehigh University has been experimenting with a college-wide introductory computer course with a considerably broader scope, combining both Fortran and numerical methods. The thinking was, and is, that this would make it possible to assign programming problems that are genuinely challenging and interesting and that represent rather realistic examples of engineering analysis in a variety of fields. The goal was to take advantage of the eagerness of able freshmen engineers to learn about engineering and computing. The course that has evolved is sufficiently unique and, we think, successful that our experiences may be of interest to others.

GENERAL COURSE DESIGN

The current catalog description for our freshmen course is:

Engr. 1. Introduction to Engineering (3) Fall-Spring Introduction to the solution of engineering problems through the use of the computer.

Elementary computer programming in Fortran is taught and illustrated by means of several topics in computational mathematics such as roots of equations, matrices, least square analysis, numerical integration, and others. No previous knowledge of computer programming is assumed.

Also, a series of lectures...

The last part of the course description deals with engineering orientation and is not germane to this paper.

A brief description of the student who is taking this course is appropriate. The average first-year engineering student at Lehigh has a 670 SAT mathematics score and a 550 SAT verbal score. About 75% of the 500 freshmen rank in the top fifth of their secondary school graduation class. Nearly a third have had some computing experience prior to arriving at Lehigh, and of these most had experience with the Basic language. Very few have had numerical methods experience.

The course is required for all engineering freshmen and students from the College of Arts and Sciences and the College of Business and Economics occasionally elect it. Finally, all students will have at least one calculus course concurrently or previously.

Each week, for 14 weeks, the course meets for two 50-minute lectures, and one 75-minute recitation period. The lecture is given to 275 students by a senior professor who has responsibility for the course organization. Recitation sections have 30-35 students and are usually led by a professor. In addition, five graduate teaching assistants are used for consultation with students, grading, and record keeping. Overall administration of the course lies within the Dean's office.

Until the summer of 1981, a CDC-6400 computer was used in the course with all work being done by batch submission. A CDC Cyber 720 computer and terminal additions in July 1981 made it possible to do the course interactively, although some minimum instruction on batch submission is given. There is about one terminal for every ten students enrolled in the course. Some students have their own terminals or join together for rental units located within their living areas. The University terminals are available in a site convenient to student housing and are available on a twenty-four hour basis. A reservation system for terminal use has not been employed.

A computer-use authorization is opened for the course and accounting procedures, transparent to the user, monitor the fiscal details. About $200 per student is budgeted for computing with additional funds being available if needed. Each student is authorized record block size to allow for four permanent files. They are responsible for management of their files and a weekly check is made to purge routinely the oldest file when the authorized limit is exceeded.

COURSE ORGANIZATION

Table I lists the course syllabus. About half the periods are devoted to Fortran principles and applications. The remaining portion of the course is spent on an introduction to some numerical methods with the application of Fortran to them.

For the past six years, three different Fortran texts have been used. Many texts are available and our choice has been made primarily on the basis of use of the text as a self-teaching instrument. We currently use, Programming with Fortran BY Seymour Lipschutz and Arthur Poe, published by McGraw Hill Book Company.

Finding a suitable text for introductory numerical methods coupled with Fortran is a far more difficult task. Furthermore, our approach, at this level, is not to become involved in the mathematical development of the numerical methods but rather to have students be aware that methods exist, and for them to be able to use the methods. Thus, we have prepared our own material (3).

Reading assignments are made to coincide with the sequence of material to be presented, and problem assignments from the ends of the text chapters are also made. These problems carry little direct weight on the final grade, but are assigned primarily for enhancing understanding of the reading material which should influence greatly the final grade.

The distinguishing feature of the Fortran half of the course is that the mathematical level is aimed specifically at an able group of beginning scientists and engineers. The programming assignments are quite demanding. The numerical methods half of the course presents the standard elements of numerical analysis in a form designed specifically for first-year engineers. The ground work is thus laid for an early exposure to actual engineering analysis in a variety of engineering and science disciplines. This is important in the career decisions that the students are making at this time. The topics covered are not inherently more difficult than elementary calculus and the treatment has been expurgated to remove all non-essential elements of formalized rigor. The proofs that remain are mainly heuristic or pictorial. Most of the elements of numerical methods contained in this course are also contained in the ordinary two-year beginning mathematics program. However, the point of view is different and the problems in a calculus class are often more analytical than numerical. Solving a problem with numbers is still the most efficient learning mechanism for a great many concepts in mathematics; and besides most students at this level find it quite enjoyable.

NOVEL FEATURES OF THE COURSE

Possibly what makes this course different from many beginning computing courses is the coupling of computing principles with fairly comprehensive problems. The problem assignments are designed to challenge the student and simultaneously develop good analytical skills for problem analysis. It is recognized that the typical first-year student is unaware of much about the engineering ideas the problems are attempting to illustrate, but through the use of descriptive material and model solutions good progress towards understanding is possible.

Another feature of the course is that through an early exposure to some numerical techniques, mathematical concepts the student has already learned are re-inforced and shown to have utilitarian value. For example, students know how to find roots of quadratic equations, and they know how many roots polynomial equations have. They also know that root finding for the latter type equation is not a simple task, but through a Newton-Raphson technique, their mathematical horizons are expanded.

Although not necessarily novel, the course is demanding of students' time and they are introduced to an intellectual discipline that should have future value.

EXAMPLE ASSIGMENTS

To learn Fortran computing programs must be written! These programs should be simple enough so that the assignment can be completed in a week or so and challenging enough so that the solutions obtained are interesting and perhaps useful. It should not be feasible to obtain solutions to the complete problem with a pocket calculator or there is no point in using a large computer. Rather than waiting until the programming abilities of the student are such that he or she can tackle complicated Fortran code without assistance, the programming problems are assigned quite early on and example problems and their complete solution are given which then may be used as models for the solution of the assignments. There are typically 5 to 7 assignments per semester and each is meant to illustrate a feature of Fortran or numerical methods as well as a topical problem in some engineering discipline. Some examples follow:

Example I - A Problem Involving DO-Loops and Characteristics of Industrial Engineering

Background Information

Your organization subscribes to the common practice of allocating one typist to each department or section. A study indicated that during various work periods some typists were busy, while others were relatively idle. Also, some typists complained of heavy workload, while others did not seem to have such a problem.

This intuitively led to the conclusion that such irregularities in workloads could be smoothed through the establishment of a typing pool. The change seemed feasible since the concerned departments were located close to one another, making it convenient to pool such services.

Your background in operations research has led to the following assignment:

1. Devise quantitative measures comparing the effectiveness of the various departments.
2. Devise a quantitative measure expressing the effectiveness of using a pool.
3. Give a method for determining the number of typists in the pool.

Waiting-line analysis (queueing theory) can be applied to this situation.

Waiting is the condition that exists when a unit that has to be serviced in some way has to wait for the service facility to give service to other units; in other words, the unit joins a queue in which it awaits its turn to be serviced by the service facility. In any servicing situation, the elements of the situation are arrivals and servicing facilities (after called channels or servers). Here, the material to be typed represents units to be serviced and the typists represent servers.

The study obtained data including:

1. Distribution of arrival times (Poisson).
2. Distribution of service times (Exponential).

The data are summarized below:

Department	Arrival Rate per hour	Service Rate per hour
1	2.2	2.97
2	1.4	2.69
3	1.9	2.78
4	2.5	2.89
5	2.0	2.93

To fulfill part 1, you have decided to compute each typists utilization, the expected number of units waiting to be typed, and the expected waiting time in each queue. You have also decided to compute the mean waiting time and compare each department to this mean.

To fulfill part 2, you have decided to compute the expected queue waiting time and the percent idle time for the typist.

To fulfill part 3 quantitatively is not practical. Although the cost of typists is known, the cost per unit (of typing) waiting is very subtle and a reliable estimate is not available. Thus, we can only provide more effective information to management to enable them to balance the two conflicting measures of queue waiting time and servers' idle time.

The following symbols will be used:

λ = mean arrival rate
μ = mean server rate (per busy server)
ρ = utilization factor
c = number of servers (in the pool)
L_q = expected number of units in the queue
W_q = expected waiting time per unit in the queue
N = number of single-server queues (departments)

The following relationships apply to the single-server departments:

$$\rho_i=\lambda_i/\mu_i \qquad L_{qi} = \rho_i^2/(1-\rho_i) \qquad W_{qi}=L_{qi}/\mu_i$$

$$\overline{W}_q=\sum_{i=1}^{N} L_{qi}/\sum_{i=1}^{N} \lambda_i \quad \text{(mean waiting time)}$$

$$\Delta W_{qi}=W_{qi}-\overline{W}_q \quad \text{(difference in waiting time from the mean)}$$

$$\overline{\mu}=\sum_{i=1}^{N} \lambda_i\mu_i/\sum_{i=1}^{N} \lambda_i \quad \text{(weighted mean service rate, for all typists)}$$

$$\overline{\rho}=\sum_{i=1}^{N} \lambda_i/N\overline{\mu} \quad \text{(mean utilization factor, for all typists)}$$

The following relationships apply to the multiple-server pool:

$\lambda_m=\sum_{i=1}^{N}\lambda_i$ $\quad \mu_m=\bar{\mu}$ $\quad \rho_m=N\bar{\rho}$ $\quad W_{qc}=L_{qc}/\lambda_m$

$X_c=100(1-\frac{\rho_m}{c})$ (percentage of servers' idle time

$L_{qc}=P_o(\rho_m)^c(\rho_M/c)/c!(1-(\rho_m/_c))^2$

$P_o=1/\{(1+\sum_{n=1}^{c-1}(\rho_m)^n/n!)+((\rho_m)^c/c!(1-(\rho_{m/c}))\}$

Specific Requirements

Write a Fortran program that will:

1. Read the department data.
2. For each department print a line containing: department number $i,\lambda_i,\mu_i,\rho_i,L_{qi},W_{qi}$, and ΔW_{qi}.
3. After all departments have been processed, leave a blank line, then print one line containing: $N,\Sigma\lambda_i,\bar{\mu},\bar{\rho},\Sigma L_{qi},W_q$. Align with the above.
4. Leave two blank lines, then print a table in which each line contains: c, W_{qc}, X_c. Vary c from 4 through 11 servers. Follow with two blank lines.

Notes:

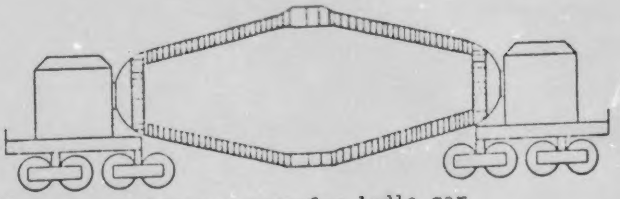
1. Your program must be able to handle up to fifty departments (use the "END=" option in the read statement).
2. Read the λ_i and μ_i into two one-dimensional arrays for later processing. Use other array(s) as you deem necessary.
3. Convert all waiting times to minutes.
4. Avoid recalculations as much as possible - use intermediate variables where necessary.
5. Use DO-loops in general, indenting and aligning the body of all loops. End all loops with a CONTINUE statement.
6. Supply appropriate column headings for all output.

Example II is a problem used to illustrate Metallurgical Engineering and to have students develop arrays and do plotting.

Example II

Background Information

The pig iron produced in a blast furnace is transferred to steel-making facilities in a ladle mounted on a railroad car. Several types of ladles are used, the most prominent one being the Pugh-type which is a cigar-shaped tilting ladle. The pig iron is tapped from the blast furnace into the ladle and then transferred to the steel-making facilities which may be some distance from the blast furnace. It is of interest to know the time which one can allow the metal to remain in the ladle before solidification begins.



150 Ton transfer ladle car

The following symbols will be used:

T_a = ambient temperature(°R)

T_m = temperature of metal at time t (°R)

T_{mn} = metal temperature after latest iteration (°R)

T_s = temperature of outside surface (°R)

ΔT = tolerance error in surface temperature (°R)

h_c = convective heat transfer coefficient

ε = emissivity of steel shell

C_p = specific heat of the metal (BTU/lb°F)

k = thermal conductivity of refractory

ω_m = weight of metal (tons)

F = factor to account for specific heat of ladle

A_0 = area of external surface (sq feet)

A_i = area of internal surface (sq feet)

$A_{\ell m}$ = log mean area for heat transfer through the refractory

ℓ = thickness of refractory (feet)

T_{sn} = surface temperature during latest iteration (°R)

Δt = time interval (hours)

Heat is transferred to the surroundings by conduction through the refractory and then by radiation and convection to the ambient surroundings at temperature T_a. The conduction heat transfer during a unit time is

$$q_{cond}=\frac{k\cdot A_{\ell m}}{\ell}(T_m-T_s)$$

where the cylindrical wall is being considered; the log mean area for the heat conduction calculation is given by the relationship

$$A_{\ell m}=\frac{A_0-A_i}{\ell n(A_0/A_i)}$$

The radiation loss during a unit time per unit area is

$$q_{rad\text{-}out}=0.173\times10^{-8}(\varepsilon)T_s^4$$

The radiation gain during a unit time per unit area is

$$q_{rad\text{-}in}=0.173\times10^{-8}(\varepsilon)T_a^4$$

By energy balance

$$q_{cond}=A_0(q_{conv}+q_{rad\text{-}out}-q_{rad\text{-}in})=q_{shell}$$

The heat loss from the metal results in a decrease in its temperature. At the start of any time interval the metal temperature T_m and the ambient temperature T_a are known. The surface temperature T_s is not known, so an initial estimate is made. This allows q_{cond} and q_{shell} to be calculated. If T_s has been estimated correctly, q_{cond} is equal to q_{shell} and the temperature of the metal at the end of the time interval can be calculated as

$$T_{mn} = T_m - \frac{\Delta t \cdot q_{cond}}{2000 \omega_m \cdot C_p \cdot F}$$

If T_s has not been determined within a small limit of error, it is necessary to correct the assumed temperature. The magnitude of the change to be made in T_s depends upon how different its most recent value is from its previous value. If

$$|T_{sn} - T_s| \geq \Delta T,$$

the new surface temperature estimate is to be

$$T_s = T_s + 100 \left(\frac{q_{cond} - q_{shell}}{q_{shell}}\right)$$

Your assignment is to calculate the temperature drop of the metal over time and plot this relationship on the line printer and plotter. You are to do this for two initial metal temperatures.

Along with the problem statement given above, some specific directions are given the students regarding constraints and details about plotting requirements.

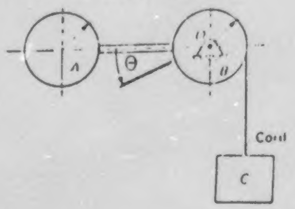
Another assignment characteristic of a problem in engineering physics is given in Example III. The main objective of this problem is to have students do programming involving a numerical method of integration.

Example III

Background Information

Pendulum systems appear in a variety of mechanisms, and can have complex motions which often do not yield to analytic solutions. An example of this is the basis of this problem.

The system below is released from rest in the position shown.



The cord attached to C winds around cylinder B as cylinder B rotates about its pivot axis O. A light rod of negligible weight connects cylinders A and B. It will be assumed that there is no friction and no viscous drag. Your assignment is to show the relationship of time and angular displacement O.

The following symbols will be used:

W = weight of each cylinder (lbs)
W_c = weight of block C (lbs)
r = radius of A and B (ft)
ℓ = distance of A to B, center-to-center (ft)
θ = angular displacement (radians)
$\dot{\theta}$ = angular velocity (radians/sec)
t = time from rest (seconds)
g = gravitational constant (32.2 ft/sec²)

By using the system angular momentum and the sum of the moments about 0, it can be shown that

$$\dot{\theta} = \frac{d\theta}{dt} = \sqrt{\frac{2}{F}(W\ell\sin\theta - W_c r\theta)} \tag{1}$$

$$\text{where } F = \frac{W}{g}(r^2+\ell^2) + \frac{W_c r^2}{g}$$

Rearranging and integrating (1) yields

$$\int_0^t dt = t = \int_0^\theta \frac{d\theta}{\sqrt{\frac{2}{F}(W\ell\sin\theta - W_c r\theta)}} \tag{2}$$

Equation (2) gives the time t for the system to rotate through an angle.

The upper limit of θ for the integral in equation (2) can be determined from the fact that $\dot{\theta} = 0$ at the end of a half cycle. Thus, from equation (1)

$$\dot{\theta} = 0 = \sqrt{\frac{2}{F}(W\ell\sin\theta - W_c r\theta)} \quad \text{or} \quad W\ell\sin\theta - W_c r\theta = 0 \tag{3}$$

The solution of Equation (3) yields the maximum value of θ(θ max).

Before equation (2) can be considered for numerical integration, it must be noted that at the lower limit (θ=0) the function is infinite. We will approximate the function between θ = 0 and θ = 1 degree using a McLaurin series for the sin function:

$$\frac{2}{F}(W\ell\sin\theta - W_c r\theta) = -\frac{2}{F}W_c r\theta + \frac{2}{F}W\ell\left[\theta - \frac{\theta^3}{3!} + O(\theta^4)\right]$$

It then can be shown that the time to traverse from 0 to 1 degrees is

$$t_1 = \int_0^{\theta_1} \left[\frac{2}{F}(W\ell - W_c r)\theta\right]^{-\frac{1}{2}} \left[1 + \frac{W\ell}{W\ell - W_c r}\left(\frac{\theta^2}{12}\right)\right] d\theta$$

or

$$t_1 = \frac{2\sqrt{\theta_1}}{\sqrt{\frac{2}{F}(W\ell - W_c r)}} + \frac{2}{60}\,\frac{W\ell}{(W\ell - W_c r)}\,\frac{\theta_1^{5/2}}{\sqrt{\frac{2}{F}(W\ell - W_c r)}} \tag{4}$$

where $\theta_1 = \Pi/180$

We can now combine equations (2) and (4):

$$t = t_1 + \int_{\theta_1}^{\theta} \frac{d\theta}{\sqrt{\frac{2}{F}(W\ell\sin\theta - W_c r\theta}}$$

Specific Requirements

Write a Fortran program that will:

1. Read and echo-print the system parameters:

$W = 32.2$ lbs $r = 0.5$ ft
$W = 16.1$ lbs $\ell = 2.0$ ft

2. Calculate Θ_{max} from equation (3) using subroutine NEWTØN with EPS = 1.E-10 and ITMAX = 50. A logical value for the initial estimate X_1 is π, since we know that $\Theta_{max} \leq \pi$. Print Θ.
3. Calculate and print T_1 using equation (4).
4. Calculate the time t_{max} required to reach Θ_{max} using equation (5) and subroutine RØMBRG with EPS = 1.E-5 and KNR = 6. Print the Romberg table and t_{max}. .
5. Produce a table of Θ and t values for $10 \leq \Theta < \Theta_{max}$ in increments of 10 degrees. Use RØMBRG to integrate, using only the interval concerned (ex::$30° \leq \Theta \leq 40°$) and summing the elapsed time. Do NOT print the Romberg tables.

Students are given three weeks to submit the assignment. When the problem is presented to the student, a faculty member from the appropriate department gives a brief presentation describing the relationship to that discipline. As can be seen from each example some background information is given the student for appreciation.

None of the problems are felt to be beyond the capabilities of the student. Consultation with classmates, graduate assistants, and the staff is permitted, but individual original solutions are expected.

Grades for the course are determined by the weightings of various aspects according to Table II. Each term students request that more credit be granted for programming assignments because of the time spent doing them. Collaboration beyond that authorized is known to take place so we are not interested in promoting more. The only yielding to this request is to grant bonus points, up to 5, for early submission.

Table II

Factors and Weightings for Final Grade

Hour quizzes	3 x 100	300
Final exam		300
Short tests in recitations	4 x 20	80
Written homework		70
Programming assignments	5 x 50	250
		1000

DISCUSSION

At the end of each semester students are urged to participate in a course evaluation. We attempt to impress upon the students that their responses are indeed used to improve the course. A variety of mechanical procedural and teacher evaluation items are included in the survey instrument, but only the items appropriate to this paper will be discussed.

Programming Assignments

Students generally feel that the programming assignments are quite useful as an instrument of learning Fortran coding, but that a typical assignment required eight to twelve hours from initiation to completion. Spread over three weeks this amount of time does not appear excessive to the staff, nor does it really appear so to the student who did not wait until the last minute to do the work. Unfortunately, some pertinent material needed for the assignment was not always available at the time when the assignment was made. So in reality all of the days within the three-week span were not always available.

In pursuing with students why so much time was needed to complete the assignment, frequent responses included:

- Waiting for a terminal
- Typing errors
- Logic errors
- General debugging
- Lack of concentration/serious effort
- Never had computing before

Further exploration of student opinion indicated that the course was a good mixture of mathematics and computing and that combination was a useful way to learn some mathematics.

Quizzes

Quizzes are designed to test basic understanding of computing principles and the numerical methods. Questions on quizzes are a straightforward reflection of course highlights so that a student who individually does the assignments and understands them has little difficulty with the quizzes.

Grade distribution in the course has been typically: 17%A, 46%B, 26%C, 10%D and 1%F. About 2% of the starting group dropped the course by the 9th week deadline.

CONCLUSIONS

Although the course is rather expensive with respect to teaching personnel and computer costs, we feel that it is a good investment. Students do leave the course with some skills that are useful in subsequent courses because these follow-up engineering courses rely heavily on the use of computer solutions for problem analysis.

ACKNOWLEDGEMENTS

A course such as is described simply does not happen without the work of many individuals. In particular we are grateful to the work of the following course lecturers who really organized the effort over the years: Bill Schiesser, Bob Johnson, Andy Kreutzer, and Ken Tarby.

REFERENCES

1. *Lehigh University Catalog 1981-1983*, Lehigh University, Bethlehem, PA.
2. Borse, G. J., *Numerical Methods Notes*, Lehigh University, Bethlehem, PA 1980.

Table I
Course Syllabus for Engineering I

Lecture	Topic
1	Course description and requirements; key punch and intercom; card deck organization, real and integer numbers/variables.
2	Integer/real arithmetic, assignment statements, arithmetic expressions and hierarchy of operations.
3	Free format I/O, system functions, mixed mode, formatted I/O,READ( ),WRITE ( ),PRINT.
4	F,I,X,H formats, control statements, GO TO, logical IF, BLOCK IF, logical statements.
5	Computed GO TO, arithmetic IF, statement function, summation, series, printer carriage control.
6	Interactive procedures, sign on, files, operational modes, INPUT, RUN, PRINT.
7	E-format, roots by bisection.
8	Interactive, EDIT mode, operational details.
9	Hour Quiz #1.
10	Dimensional variables, internal arrays, sums, DO-loops, E-notation.
11	Nested DO-loops, series summation, sorting and arranging, implied DO-loops.
12	A-format, functions and sub-routines, argument lists, rules.
13	External statements, labeled and blank COMMON.
14	Cal-comp plotting, scaling.
15	Numerical methods introduction, derivatives, integrals, Taylor/MacLaurin series approximation.
16	Hour Quiz #2.
17	Taylor series, convergence, errors, use of series.
18	Roots of equations, review bisection, Newton, Raphson.
19	Multiple roots, modified Newton, numerical differentiation, scant method.
20	Linear algebra, matrices, terminology, operations, Cramer's rule.
21	Singular, ill conditioned systems, Gauss-Jordan elimination, prints, maximization of prints.
22	Fortran for Gauss-Jordan, inverse by Gauss-Jordan, library sub-routines.
23	Gauss-Seidel iteration, lease squares curve fitting, least squares parabola.
24	Hour Quiz #3.
25	Fortran for least squares, introduction to numerical integration, trapezoidal rule.
26	Simpson's rule, Cote's rule, relationships leading to Romberg integration.
27	Fortran for Romberg.
28	Numerical methods and rounding errors.
29	Three-hour final examination.

NATURAL LANGUAGE COMPUTING IN A LINEAR ALGEBRA COURSE

Robert Geist
Department of Computer Science

David Kraines
Department of Mathematics

Pamela Fink
Department of Computer Science
Duke University, Durham, North Carolina 27706
919-684-3048

§1 INTRODUCTION

Linear Algebra is the branch of mathematics concerned with the theory of many equations in many unknowns. The subject is applied regularly in most of the natural sciences, to a great extent in Economics, and to some extent in Psychology and Sociology. More than 25% of the undergraduates at our University are now exposed to this material.

Many problems of interest in Linear Algebra involve extensive arithmetic computations. In the past, artificially simple problems were given to the students so that they would not spend an undue amount of time on the tedious, but unenlightening computations. In recent years, the simultaneous arrival of affordable digital computers and a "new breed" of application-oriented students [4] has made more realistic problem assignments at once both feasible and necessary.

There are however still difficulties. Either the student is required to write his own software, a time-consuming effort for most students, or the student is given a "canned" program which does too much, that is, eliminates the need for the student to truly understand the problem-solving process under consideration.

In this paper we present the results of our initial experiment with a proposed solution to such difficulties: student use of the Natural Language Computer (NLC), a software system developed by Ballard, Biermann, Fink, and Holler (see [1], [2], [3], and [5]) to demonstrate the feasibility of a structured approach to implementing natural English as a programming language. In brief, NLC is an interactive system which accepts imperative English sentences as commands to manipulate a user-defined world of matrices. This "matrix world" was chosen over others (e.g. Winograd's blocks [9]) as one in which real-world problems could be solved and in which there existed a natural and simple semantic hierarchy (matrices contain rows and columns which contain entries, etc.).

Nevertheless, the system, described in Section 2, is not a Linear Algebra system. For example, the command

"Multiply B by C."

referring to two equal-sized matrices named B and C, would yield a naive term-by-term multiplication (with answer left in B). Thus a matrix multiplication in the standard Linear Algebra sense, a completely different operation, would have to be carried out either in a step-by-step fashion, in which the system functions as a vector-based desk calculator with English interface, or through the system facility for allowing the user to define his own imperative verbs in terms of previously understood concepts.

Our initial goal for the project was closely tied to this facility for user-defined imperatives. Specifically, it was hoped that students, forced to express their understanding of algorithms in unambiguous English, would develop a true mastery of the required problem-solving processes, without the time-consuming digression into computer programming. As detailed in Section 4, the results of the experiment allow us to claim a qualified success in achieving this goal.

We conclude this paper by presenting a summary of student reactions in Section 5 and our conclusions and current directions in Section 6.

§2 NLC

The Natural Language Computer (NLC) used in this experiment is an imperative sentence processor that works on the matrix domain. It analyzes typed English commands and then performs the requested operation(s) on the matrix world displayed on the screen. Thus, the user can see immediately whether or not the command was interpreted correctly.

The following illustrates the basic imperative sentence structure accepted by NLC:

Imperative Verb (Noun Group) (Noun Group) (Noun Group)

The number of noun groups following the imperative verb is determined by the number of operands required by the given imperative verb. For example, the verb 'backup' requires no operands. On the other hand, 'double' requires one, as shown in the following sentence:

Double row 1.

The verb 'add' requires two operands as shown below:

Add row 1 to row 2.

or possibly three operands as in the following:

Add row 1 to row 2 putting the result in row 3.

The structure of these operands, or noun groups, can become quite complicated. The basic noun group structure accepted by NLC is as follows:

(determiner) (ordinal) (number) (adjective)* (classifier)* noun (qualifier)*

where * indicates closure under concatenation. The only non-optional element in the entire phrase is the noun itself. Thus, NLC can process a noun group as complex as the following:

the last two even matrix 1 entries that are negative

where every possible member of the noun group structure occurs, or as simple as

entry 2,3

where only the required element, a noun, occurs.

Certain variations on the basic imperative sentence are allowed in NLC. These are the addition of conditional and looping clauses to the beginning or end of the command. These clauses are called control clauses because they control whether or not and how many times the command that they modify is executed. Thus, the most general, top-level structure of the sentences accepted by NLC is as follows:

(Control Clause)* Imperative Sentence (Control Clause)*

The structure of the control clause varies depending on the type. The structure of a conditional control clause resembles closely that of a noun group consisting of a noun and a qualifier. The main difference is that in a conditional clause, the qualifier is actually the predicate of a declarative sentence. Thus the noun group

the rows containing a positive entry

becomes

if row 1 contains a positive entry.

Certain looping clauses have a structure similar to that of the conditional clause. The only difference is that the head conditional word such as 'if' or 'unless' becomes a head looping word such as 'while' or 'until'.

The user is also allowed to define new imperative verbs in NLC. Such a definition creates, in effect, a subroutine that can be invoked by simply using the newly-defined verb in a command. In defining a new verb, a special format must be followed:

```
   Define a way to newverb (operand-1
(preposition operand-2)).
   [  procedure
   [    body
    end.
```

The first line of a verb or procedure definition must always be of the form shown in the first sentence above. The new verb (the name of the procedure), in this case 'newverb', must be given, as well as the operands that it requires. The number of operands may vary from zero to two, depending on the needs of the verb being defined. However, they must be indicated by names, such as 'operand-1' or 'cat', and never by the actual element, such as 'matrix 1' or 'entry 2,3'. This is because the operands in this initial sentence are the arguments to the procedure and thus must be variables, for they will be different each time the procedure is invoked.

The procedure body can be any legal series of NLC commands. As each command is typed, NLC interprets it and performs the requested operation, using the named operands as context. In this way, the user can check the correctness of the procedure step-by-step. The end of the procedure definition must be indicated by the command 'end' or 'end of procedure'.

S3 THE EXPERIMENT

NLC is written in the C programming language and now runs under the UNIX operating system on the Computer Science Department's PDP 11/70. This minicomputer is rather heavily used by graduate students and faculty, with a typical load of eight users doing program development and text editing. Since all modules of NLC together occupy roughly 260K bytes, a significant portion of the main memory space available for swapping, it was estimated that no more than three or four undergraduate users could be supported by the present system without seriously degrading total performance.

Thus, in the Fall of 1981, a section of Mathematics 106, Linear Algebra with Digital Computation, was presented with the option, for a limited number of students, to use NLC in their course work, rather than the collection of packaged programs developed by Fraleigh and Frederick for CONDUIT [6]. Response was too good. Six students, one quarter of the class, were selected solely on the basis of interest, and were given an introductory demonstration of the system. Each was also given a brief written tutorial on using the system [7], and a Beehive 150 terminal, having the capability of fast screen updates through its cursor-addressing, was placed on reserve for their exclusive use.

The textbook for Mathematics 106 (Bradley [4]) was chosen independently of NLC considerations. The NLC students, with some faculty assistance, had to develop and use three algorithms: matrix multiplication, row-echelon reduction, and Gram-Schmidt orthogonalization, all described in the next section. These were key to the course in that various modifications of these three could provide the student with a nearly complete repertoire of linear algebra problem-solving tools. The students were presented with an NLC version of one of the three, row-echelon reduction, as a non-trivial example of user-defined imperatives. Since this algorithm was needed early in the course, it also provided a "running start".

Dialogues of student-NLC interactions were recorded by the system in files which permitted no student access. The students were informed, however, that their interactions were being recorded and reviewed.

S4 RESULTS

As was expected, the students were most successful (and happiest!) when operating the system in "calculator mode." As an example, here is a dialogue used to find the so-called inverse of the matrix $\begin{pmatrix}2 & 5\\ 3 & 7\end{pmatrix}$, in which each typed command is followed by the subsequent CRT display:

```
Create a 2 by 4 matrix.
  1  0.00  0.00  0.00  0.00
  2  0.00  0.00  0.00  0.00

Put 2,5,1 and 0 in row 1.
  1  2.00  5.00  1.00  0.00
  2  0.00  0.00  0.00  0.00

Put 3,7,0, and 1 in row 2.
  1  2.00  5.00  1.00  0.00
  2  3.00  7.00  0.00  1.00

Divide row 1 by 2.
  1  1.00  2.50  0.50  0.00
  2  3.00  7.00  0.00  1.00

Subtract 3 times row 1 from row 2.
  1  1.00  2.50  0.50  0.00
  2  0.00 -0.50 -1.50  1.00

Divide row 2 by -0.5.
  1  1.00  2.50  0.50  0.00
  2  0.00  1.00  3.00 -2.00

Subtract 2.5 times row 2 from row 1.
  1  1.00  0.00  -7.00  5.00
  2  0.00  1.00   3.00 -2.00

Quit.
```

Student-defined imperatives were another matter. We consider the three key algorithms separately.

4.1 Matrix Multiplication

Recall that if X is an m by n matrix and Y is an n by p matrix then their product Z=X·Y is an m by p matrix whose $(i,j)^{th}$ entry, z_{ij}, is given by

$$z_{ij} = \sum_{k=1}^{n} x_{ik}\cdot y_{kj}$$

The students were entirely successful in defining this procedure for themselves in NLC. Here is a representative student dialogue:

```
Create a 2 by 3 matrix.
Call it X.
Create a 3 by 2 matrix.
Call it Y.
Create a 2 by 2 matrix.
Call it XY.
Create a 1 by 3 matrix.
Call it aux.
Create i.
Create j.
Call entry 1,1 of XY z.
Define a way to innerprod z.
   Store the number of the row containing
   z in i.
   Store the number of the column
   containing z in j.
   Multiply row i of X by column j of
   Y putting the answer in aux.
   Place the sum of the entries of aux in
   z.
   end.
Unname z.
Define a way to compose XY.
   innerprod all entries in XY.
   end.
Quit.
```

To use the defined imperative, 'compose', thereafter, the student need only name his operands X and Y, create an answer matrix of the appropriate size, and issue the command 'compose' followed by the name of the answer matrix. The dimensions of X and Y need not be 2 by 3 and 3 by 2! Though the definition process requires specific operands, the definition itself (and subsequent use) need not depend upon any particular parameters of the specific operands being used in the defining process.

More sophisticated definitions were certainly possible. Nevertheless, dialogues such as the one above were considered entirely acceptable, and imperative-definition seemed well within student capability at this level of complexity.

4.2 Row-Echelon Reduction

A matrix is said to be in row-echelon form provided:

i) The first non-zero entry in each row is a 1.
ii) The columns containing row-leading 1's have only 0's below these 1's.
iii) The row-leading 1's descend to the right.

The procedure of transforming an arbitrary matrix into row-echelon form is absolutely essential in Linear Algebra: it is the key step in solving systems of linear equations by the method known as Gaussian elimination. It plays an equally central role in computing an important square matrix characteristic, called the determinant, and can also be used to compute the inverse of a matrix, when one exists. (Recall, two square matrices A and B are said to be inverses of one another if their product is the identity matrix, that is, a square matrix with 1's on the diagonal and 0's elsewhere.)

In the first week of class the students tried to produce an NLC version of this procedure on their own. No one was successful, and it was deemed necessary to provide them with the following solution:

```
Create a 3 by 4 matrix.
Fill it with random numbers.
Call it Duke.
Create i.
Store 1 in i.
Call entry 2,1 of Duke z.
Define a way to Scrub z.
     Call the row containing z x.
     Subtract the product of z and row i
     from x.
     Unname x.
     end.
Unname z.
Call column 1 of Duke y.
Define a way to klear y.
     Consider the first non-zero entry in
     y not above the i-th entry in y.
     Exchange the row containing that
     entry with the i-th row.
     Divide row i by the i-th entry in y.
     Scrub all entries in y below the i-
     th entry in y.
     Add 1 to i.
     end.
Define a way to reduce y.
     If the i-th entry in y is non-zero
     or an entry below the i-th entry in
     y is non-zero then klear y.
     end.
Define a way to Check y.
     If the number of the last row in
     Duke is not less than i then reduce
     y.
     end.
Unname y.
Delete i.
Define a way to Gauss Duke.
     Create i.
     Store 1 in i.
     Check all columns in Duke.
     end.
Quit.
```

To use this procedure on any matrix thereafter, the student need only name his matrix Duke and issue the command, 'Gauss Duke'.

Though this procedure is not significantly longer than the matrix multiplication, it is certainly more delicate. In particular, the use of the auxiliary column indicator, i, to mark "progress made" may very well be unnatural to those with limited experience in algorithm development.

All students were subsequently able to use this 'Gauss' imperative in calculator mode to solve systems of equations and compute matrix inverses and determinants. Unfortunately, not all were able to define their own imperatives to carry out these extensions of the basic 'Gauss' imperative. This was a disappointment in that we had hoped such extensions would be within the capability of every student.

4.3 Gram-Schmidt Orthogonalization

Any square matrix M with non-zero determinant can be factored into two matrices of the same size as M,

$$M = Q \cdot R$$

so that the columns of Q are all vectors of length 1 which are orthogonal to one another, and R has only 0's below its diagonal.

This Gram-Schmidt orthogonalization process is quite important in a number of applications. One of the most intriguing of these is a method for computing the so-called eigenvalues of M. A number is called an eigenvalue of M if there is some non-zero vector X for which $M \cdot X = \lambda X$. It turns out that if we factor M by the Gram-Schmidt process, $M = Q_0 R_0$, then let $M_1 = R_0 Q_0$ and repeat $M_1 = Q_1 R_1$, $M_2 = R_1 Q_1$, etc., the sequence M_i will, under very general conditions, converge to a matrix having the eigenvalues of M on its diagonal [8].

After a study of the theory of eigenvalues, the students were asked to develop, as a final project, an NLC imperative to carry out this QR algorithm for finding eigenvalues. Two were almost completely successful, the others made substantially less progress. The total process is, admittedly, rather intricate.

The key step, of course, is the factorization, which may be carried out as follows:

```
Create a 3 by 3 matrix.
Fill it with random numbers.
Call it M.
Create a 3 by 3 matrix.
Fill it with random numbers.
Call it Q.
Create a 3 by 3 matrix.
Call it R.
Create i.
Store 1 in i.
Create a 1 by 3 matrix.
Call it AUX.
Call column 1 of Q v.
Define a way to influeremove v.
    Create coeff.
    Multiply column i of M by v putting
    the result in row 1 of AUX.
    Store the sum of the entries in  AUX
    in coeff.
    Multiply v by v putting  the  result
    in row 1 of AUX.
    Divide  coeff  by  the  sum  of  the
    entries in AUX.
    Subtract the product of coeff and  v
    from column i of Q.
    Create j.
    Store the number of v in j.
    Store coeff in entry j,i of R.
    Delete j.
    Delete coeff.
    end.
Define a way to suborthog v.
    If the number of v is  less  than  i
    then influeremove v.
    end.
Delete i.
Define a way to orthog v.
    Create i.
    Store the number of v in i.
    Copy column i of M to v.
    Store 1 in entry i,i of R.
    Suborthog all columns in Q.
    Delete i.
    end.
Define a way to normalize v.
    Multiply v by v putting  the  result
    in row 1 of AUX.
    Create norm.
    Store the square root of the sum  of
    the entries in AUX in norm.
    Divide v by norm.
    Create i.
    Store the number of v in i.
    Multiply row i of R by norm.
    Delete i.
    Delete norm.
    end.
Unname v.
Delete AUX.
Delete R.
Delete Q.
Define a way to GramSchmidt M.
    Create i.
    Store the number of the last row  of
    M in i.
    Create an i by i matrix.
    Call it Q.
    Create an i by i matrix.
    Call it R.
    Create a 1 by i matrix.
    Call it AUX.
    Delete i.
    Orthog all columns in Q.
    Normalize all columns in Q.
    Delete AUX.
    end.
Quit.
```

Note that since all needed auxiliaries, including Q and R, are created by the procedure itself, the user need only name his matrix M and issue the command 'GramSchmidt M'.

S5 STUDENT REACTIONS

The reactions from all students were characterized by harsh micro-level criticism and glowing macro-level praise. All found the calculator mode of operation to be easy and enjoyable, so much so that several requested that all course algorithms be installed as primitive system imperatives in order to completely eliminate the difficult and frustrating problem of defining their own imperatives. Some specific complaints were:

1. No listing of recognized imperatives. Several needed to use a thesaurus as a programmer's manual.
2. Poor editing facilities. Mistakes were easy to fix in calculator mode via the 'backup' command, but changes to user-defined imperatives almost always required a complete re-typing.
3. Poor error messages. As a typical example, this sequence:

   ```
   Create a 3 by 3 matrix.
   Call it A.
   ```

   would produce the message

   ```
   UNABLE TO PARSE PAST TOKEN 3
   ```

   which fails to convey the real problem, namely that NLC doesn't distinguish between upper and lower case and does recognize as English the word 'a'. Call it <u>a</u> what?
4. Inadequate preparation for defining imperatives. The tutorial [6] was designed as a general purpose introduction to the system and concentrated on the calculator mode of operation to the exclusion of user-defined imperatives.
5. Limited screen display. Though the system properly maintained all matrix entries internally, the Beehive 150 terminal was unable to display more than 6 columns of any matrix under consideration.
6. Response time. This was only an issue on the Gram-Schmidt project, but there it was a considerable problem: a single QR factorization often took as long as 30 minutes.

In spite of these problems, the overall reaction was very positive. All indicated that they would again choose NLC over the canned program approach,

BEST COPY AVAILABLE

given the opportunity. Here are some representative quotes:

"In summary, I think that NLC would be the perfect aid for MATH 104 students to solve systems of linear equations ..."

"On the whole I feel NLC was a very worthwhile project. It was exciting working on an experimental computer ..."

"In spite of all the headaches and frustrations in dealing with NLC, I regard [this] as one of the best learning experiences that I have had at Duke."

S6 CONCLUSIONS

We have presented the results of a preliminary experiment with the Natural Language Computer in a Linear Algebra course. Hardware facilities have not allowed us the luxury of a student sample size adequate to draw definitive, statistically significant conclusions. Nevertheless, we feel that we have obtained a qualified success in demonstrating the feasibility of removing formal programming from the Linear Algebra course without removing the opportunity for a thorough understanding of the problem solving processes which are fundamental to the subject. At the very least, our basic thesis strongly deserves further investigation, and, to this end, preparations are underway for an enlarged experiment in the Fall of 1982. Specific changes now in progress are:

1. An improved tutorial.
   This will be tailored to the Linear Algebra course and emphasize the system facility for user-defined imperatives.
2. Dedicated hardware.
   Funds will be sought for several major items. A fast CPU with 32-bit addressing, dedicated to NLC, would allow us to support a sufficient student sample size and would improve response time substantially. This dedicated processor should be accompanied by at least eight terminals having 10 column matrix capability and a classroom display facility for on-line system use by the instructor during class sessions.
3. Algorithm primitives.
   All key algorithms will be installed as primitive system imperatives. This is not in contradiction to our basic thesis, but simply to allow the instructor additional flexibility in tailoring the course to his particular goals and the specific needs of his students. These additional imperatives will be easily hidden from student access, whenever such is desired.

Natural language computing is in its infancy. Driven by applications such as these, it should experience a rapid and vigorous growth to assume a position of considerable prominence in many fields, the most important of which is certainly education itself.

This work was supported in part under the National Library of Medicine Program Project Grant LM-03373 and the National Science Foundation Grant MCS-7904120.

REFERENCES

[1] Ballard, B.W., Semantic Processing for a Natural Language Programming System, Ph.D. Dissertation, Duke University, Durham, N.C., 1979.

[2] Biermann, A.W. and Ballard, B.W., Towards Natural Language Programming, American Journal of Computational Linguistics, 6(1980).

[3] Biermann, A.W., Ballard, B.W. and Holler, A.M., An Experimental Study of Natural Language Programming, Technical Report CS-1979-9, Duke University, Durham, N.C., 1979.

[4] Bradley, G.L., A Primer of Linear Algebra, Prentice-Hall, 1975.

[5] Fink, P.K., Conditionals in a Natural Language System, Master's Thesis, Duke University, 1981.

[6] Fraleigh, J.B., A Computer Supplement for Linear Algebra, CONDUIT, 1977.

[7] Sigmon, A.H. and Fink, P.K., Using the Natural Language Computer: A Short Tutorial, Technical Report CS-1980-4, Duke University, Durham, N.C., 1980.

[8] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford University Press, 1965.

[9] Winograd, Understanding Natural Language, Academic Press, 1976.

THE USE OF COMPUTERS IN A FIRST COURSE IN NUMERICAL ANALYSIS

Zane C. Motteler
Department of Mathematical
and Computer Sciences
Michigan Technological Univ.
Houghton, MI 49931
906-482-8146

ABSTRACT

This paper discusses undergraduate numerical analysis courses and makes recommendations concerning the extent to which students should be required to use large scale computing facilities.

INTRODUCTION

Throughout this paper, references to "a first numerical analysis course" or simply "numerical analysis" should be understood to mean the first course for undergraduates devoted exclusively to this topic, regardless of the level or content. Clearly numerical analysis can be, and frequently is, a component of other courses such as calculus and differential equations, but this paper discusses only the separate course.

The first course may be taught as early as the sophomore year, with only some calculus as a prerequisite, or as late as the senior year, requiring sophistication in linear algebra and differential equations. Over the past fifteen years the Committee on the Undergraduate Program in Mathematics (CUPM) of the Mathematical Association of America and the Association for Computing Machinery (ACM) have made a number of different recommendations regarding such a course ([1],[2],[4],[5],[6]), which shall be discussed in more detail in Section 3 below. These recommendations address the subject matter in considerable detail, but, although they clearly recommend the use of computers, do not discuss this nearly so much.

The purpose of this paper is to discuss specifically the degree of computer usage which ought to be required in numerical analysis, "computer" meaning a large scale modern electronic digital computer. Most colleges and universities of modest size or larger either have such computing power on campus or have access to it through intercampus computing nets. While one can do numerical analysis algorithms on microcomputers or even programmable calculators, the large-scale computing environment allows far more latitude in the length and complexity of algorithms, and will be the only one discussed here.

THE SCOPE OF THE PROBLEM

The primary question to be addressed is the extent to which numerical analysis is a computing course (meaning that it involves or imparts programming skills, algorithm design, the actual use of computers, etc.), as opposed to the extent to which it is a mathematics course (meaning that it involves rigorous mathematical definitions, derivations, proofs, and analysis). Given that it is a course which is taught to a wide variety of students with different interests, how should it be taught?

Many mathematicians with minimal or no computing experience teach a very theoretical course, which studies numerical algorithms and their derivation and convergence properties, with little or no consideration of their practicality. In contrast to this noncomputational approach is a course described by Cline [3], which is essentially a course in advanced programming techniques using numerical analysis examples. Even computer-oriented courses may differ strikingly: Gazzaniga and Ironi [8] describe a numerical analysis course in which students use computer-aided instruction (CAI) to study the behavior and properties of preprogrammed numerical anlaysis algorithms but apparently produce none of their own.

These differences in the computer emphasis in numerical analysis courses are reflected in the available textbooks. Some, such as Ralston and Rabinowitz [11], are mathematically rigorous and could be used in a theoretical course. No explicit computer algorithms are included in the text. The same is true of Hornbeck [9], but the latter is not rigorous and is more oriented toward computation, with many solved problems in the text. Many of the exercises are explicitly for computer solution. Conte and deBoor [7] and Moursund and Duris [10] give many algorithms in Fortran (unfortunately, in obsolete versions). Smith [12] gives algorithms in Fortran and even Basic(!).

It is clear from this discussion that the appearance of "numerical analysis" on a student's transcript tells us very little about the student's actual experience. While we may (probably correctly) assume that the student knows something about interpolation, approximation, finding roots of equations, quadrature, and the like, we know neither the extent to which the student understands such theoretical concepts as convergence, stability, and error analysis, nor whether the course was devoid of computation, used it passively (CAI, "canned" programs), or required substantial programming effort.

The substantial programming effort simply ought to be there. Students are not well served if they see numerical analysis in isolation from its most valuable tool, a modern large-scale computing environment.

HISTORICAL DEVELOPMENT OF NUMERICAL ANALYSIS

Prior to the mid-1950s, numerical computations were limited to what could be done by hand on mechanical desk calculators, and many of these computations, simple by today's standards, were lengthy and tedious. Numerical analysis courses were limited to teaching these few techniques and theory. Such courses were not widely taught and the subject matter was not always considered "respectable."

With the coming of the computer, numerical analysis began a rapid development which continues today. Old algorithms and procedures became obsolete because they were impractical or because of new techniques and algorithms which had not previously been feasible. Error analysis and stability questions became much more important as the length and complexity of tractable calculations grew. By 1965, many campuses were able to offer numerical analysis, and the CUPM included in *A General Curriculum for Mathematics in Colleges* [4] a recommendation for Mathematics 8, Numerical Analysis. The list of topics included many which sound familiar today. More noteworthy is that the course was to be "oriented towards machine computation." ([4], p. 16.) How and to what extent was not spelled out; furthermore, the subject matter outline did not specifically suggest a discussion of numerical error.

Curriculum 68 [1], ACM's first detailed curriculum recommendation, specifies three courses. The first, Course B4, Numerical Calculus, was intended to "afford [the student] an opportunity to apply the programming techniques he has learned in Course B1." ([1], p. 158.) The recommendation states flatly that "many of the standard elementary numerical analysis courses now offered in *mathematics departments* cannot be considered as substitutes for this course." (*Ibid*.; italics mine). The outline of Course B4 includes a unit on "basic concepts of numerical error." Curriculum 68 also contains recommendations for a two term sequence in numerical analysis, Courses I8 and I9, which "are intended to be mathematically rigorous and at the same time computer-oriented." ([1], p. 159.) The subject matter has been divided so that Course I9 contains all the linear algebra-related material, a division always subsequently followed by ACM. In each course, "selected algorithms will be programmed for solution on computers," (*Ibid*.) and the courses are to emphasize "efficiency, accuracy, and suitability for high-speed digital computing." ([1], p. 184.)

Thus the ACM early recognized that many numerical analysis courses taught by mathematicians did not contain adequate provision for computing. They proposed first that actual programming of algorithms be a part of the course (without, however, specifying how large a part), and second, that the methods discussed be apt for computer solution.

CUPM followed in 1971 with *Recommendations for an Undergraduate Program in Computational Mathematics* [5], which includes CM2, Introduction to Numerical Computation (similar to Course B4), and CM4, Differential Equations and Numerical Methods (topics from I8 and I9). They propose stressing "the analysis of roundoff and discretization errors, as well as the efficiency of algorithms," ([5], p. 7) and specify that "students should solve a number of problems on the computer" ([5], p. 25) not only by programming themselves, but also by making use of library subroutines.

In the following years, ACM continued its study of curriculum, eventually producing its second major recommendation, Curriculum '78 [2]. Although Course B4 of Curriculum 68 appears to have been assimilated into others, Courses I8 and I9 reappear with some changes as CS17 and CS18. These courses are to treat not only the "basic algorithms," but also their "theoretical foundations." ([2], p. 159). In particular, "programming assignments should be given to illustrate solutions of realistic problems rather than just the coding of various algorithms." (*Ibid*.) About 15% of the time is to be devoted to floating point arithmetic, finite precision, and roundoff error, and another 5% to the use of mathematical subroutine packages.

The most recent word on numerical analysis is from CUPM in *Recommendations for a General Mathematical Sciences Curriculum* [6], which proposes "four or five computer exercises for each major topic," ([6], p. 32) with groups of students cooperating on larger projects. Unlike Curriculum '78, this emphasizes that "the concept of utilizing 'canned' programs ... should be stressed" because it "does not overemphasize the efforts necessary to program a problem." (*Ibid*.) CUPM would seem to anticipate that students' programming skills might be such as to preclude major projects for fear they would bog down in details.

Nevertheless, the major professional organizations in mathematics and in computer science agree that computer usage is an essential part of a numerical analysis course. Neither group has attempted to specify this usage in as much detail as it has prescribed course content, however.

RECOMMENDATIONS

There is considerable agreement among professionals on the appropriate subject matter of numerical analysis. Thus the first recommendation is non-controversial:

Recommendation 1. The subject matter of the first undergraduate course or sequence in numerical analysis should be an appropriate subset of Curriculum '78's CS17 and CS18 [2] or the CUPM's Numerical Analysis Course [6].

Next we address the extent to which "canned" programs or subroutine libraries ought to be utilized, as some ACM and CUPM proposals suggest. This author's experience is that the great majority of subroutine libraries available today have multiple defects: they are inadequately and sometimes incorrectly documented; they contain minor (sometimes major) errors and imperfections;

and many of them were written ten to twenty years ago, so that frequently the programming is inefficient and unstructured, uses an obsolete version of a language (e.g., Fortran II), or does not use the best or most current known version of an algorithm. Thus a likely more controversial recommendation:

Recommendation 2. The use of "canned" numerical analysis routines should be minimized except possibly as exercises in how to update obsolete code.

Some "canned" programs, of course, can be of considerable help. For example, in problems involving approximation of functions, interpolation, solution of differential equations, and the like, it is frequently helpful to produce graphical output. Thus, for installations which have the capability,

Recommendation 3. Students should be instructed in the use of existing graphics packages (if available), and should be assigned at least one programming project in which the output is required in graphical form. (I propose this knowing full well that "canned" graphics packages are subject to the same deficiencies as any others. Not only that, but graphics hardware seems to be unusually susceptible to breakdown. Perhaps by instituting such a requirement we can eventually induce university computing centers to cure the problem).

The same comments as above apply to the use of CAI in a numerical analysis course. The use of CAI implies the existence of pre-programmed numerical analysis algorithms (as well as, possibly, interactive graphics, depending on the installation). Any such interactive instructional packages had better be extremely polished, robust, and absolutely foolproof. If there is a bug in them anywhere, some student will find it, possibly destroying the instructor's credibility. (It was the author's sad fate as a student, more than once, to discover flaws in my instructors' CAI programs). Thus

Recommendation 4. Do not require the use of interactive or CAI programs demonstrating numerical analysis techniques unless several months of heavy use have demonstrated such programs to be substantially error free.

CUPM and ACM definitely want the students in numerical analysis to gain experience in actually programming and testing their own algorithms. The background of current students in numerical analysis courses almost invariably includes at least one programming course, if not more. (In the last few years, the author has observed a remarkable increase in programming ability among his students in numerical analysis. Only the first Fortran course is a prerequisite, but most students have had either other courses or programming experience substantially in excess of this minimum).

It has already been noted above that many textbooks contain computer algorithms for certain numerical analysis procedures. Many of these are couched in obsolete versions of Fortran (arithmetic IF's, no END option on READ), are poorly structured (IF-GOTO exit from a DO loop), or are in an inappropriate language for serious numerical analysis (Basic). In addition to these flaws, it is not clear that these are superior in any way to "canned" programs. One has to take it on faith that they are correct and that they use the advertised technique. Thus

Recommendation 5. Avoid the temptation to have students copy and use programs given in textbooks. These may be quite useful for comparative study, however, after a student has prepared his or her own program.

In numerical analysis, there is simply no substitute for experience. Passive use of "black box" programs does not impart any useful information on the reasons for the choice of one algorithm over another nor on the difficulty of implementing one in comparison with another. These final recommendations embody this observation:

Recommendation 6. Numerical analysis should be taught by a professor who is a competent programmer familiar with the behavior of numerical algorithms on computers.

Recommendation 7. Programming projects in a numerical analysis course should be done in an appropriate language such as Fortran 77, Algol, or PL/I, as opposed to Cobol or Basic.

Recommendation 8. Approximately 50% of the grade in a numerical analysis course should be based on student programming projects implementing some of the algorithms studied in the course.

Recommendation 9. The student programs in the course should include, but not necessarily be limited to, the following:

a. Simple algorithms--e. g., interpolation, an integration technique (one week to complete)

b. Programming and comparison of two or more simple algorithms (two weeks or more)

c. More complex projects such as least squares, matrix eigenvalues, orthogonal or nonlinear least squares, etc. (four or more weeks)

d. Group projects, where carefully selected groups design algorithms together (five or more weeks)

These recommendations, of course, represent a point of departure for discussion. It is always possible that there ought to be separate numerical analysis courses for mathematicians, engineers, and computer scientists, in which the mathematical rigor and the computer content vary according to the clientele. Over the last several years, however, the author and his colleagues have taught a senior-level numerical analysis course similar to CS17, embodying the above recommendations. The student body of the course is quite varied, including undergraduate mathematics, computer science, and engineering students, as well as a sizable minority of engineering graduate students. The course is a popular and successful one, as well as being quite demanding. All students learn a significant amount of new mathematics, and many experience a substantial increase in programming skills as well. These ought to be the goals of any numerical analysis course.

BIBLIOGRAPHY

[1] Association for Computing Machinery Curriculum Committee on Computer Science, Curriculum 68: Recommendations for Academic Programs in Computer Science, Comm. ACM 11 (1968), pp. 151-197.

[2] __________, Curriculum '78: Recommendations for the Undergraduate Program in Computer Science, Comm. ACM 22 (1979), pp. 147-166.

[3] Cline, A. K., A course on mathematical software, SIGNUM Newsletter, Oct. 1981, p. 25.

[4] Committee on the Undergraduate Program in Mathematics, A General Curriculum for Mathematics for Colleges, Mathematical Association of America, Berkeley, 1965.
[5] ________, Recommendations for an Undergraduate Program in Computational Mathematics, Mathematical Association of America, Berkeley, 1971.
[6] ________, Recommendations for a General Mathematical Sciences Program, Mathematical Association of America, Washington, 1981.
[7] Conte, S. D. and C. de Boor, Elementary Numerical Analysis: an Algorithmic Approach. Third Edition. McGraw-Hill, New York, 1981.
[8] Gazzaniga, G., and L. Ironi, A CAI course of exercises in numerical analysis, SIGCUE Bulletin, Vol, 14, No. 2 (April, 1980), pp. 2-12.
[9] Hornbeck, R. W., Numerical Methods. Quantum, New York, 1975.
[10] Moursund, D. G., and C. S. Duris, Elementary Theory and Applications of Numerical Analysis. McGraw-Hill, New York, 1967.
[11] Ralston, A., and P. Rabinowitz, A First Course in Numerical Analysis. Second Edition. McGraw-Hill, New York, 1978.
[12] Smith, W. A., Elementary Numerical Analysis. Harper and Row, New York, 1979.

INTRODUCTORY QUANTUM MECHANICS USING MICROCOMPUTERS

K. S. Min
East Texas State University
Commerce, Texas 75428
(214)886-5483

Don Kosterman
Paris Junior College
Paris, Texas 75460
(214)785-7661

1. INTRODUCTION

Recently, we attempted to introduce new teaching techniques in undergraduate and graduate quantum mechanics instruction to enchance the effectiveness of the teaching and stimulate the students' interest.

Some of our objectives were:

1) In addition to the conventional analytical presentation, provide a realistic illustration of problems by utilizing a dynamic graphical description of a quantum mechanical system to stimulate students' interest.

2) Taking full advantage of high resolution graphics capabilities of some microcomputers, simulate and animate the quantum mechanical system under consideration to assist full understanding of the problem by students.

3) In every stage of this mode of instruction, interactive programs are designed in such a way that the student will select the appropriate quantum numbers and other parameters so the he/she will "construct" and "control" a quantum world of his own selection.

4) Some abstract concepts and mathematical properties of special functions will be displayed and illustrated by high resolution graphics capabilities to facilitate easier comprehensions by students.

5) Make a room for students to develop their own ideas and concepts by extending and supplementing the program, thereby they can have opportunity of learning simulation and animation techniques.

These objectives were fulfilled by developing 4 major program modules for Apple II and TRS-80 color computers available on our campuses. Necessary tool kits for scientific symbols and mixed mode of operations of text with graphic displays were developed by one of us (D.K.).

Section 2 presents a brief description of modules developed for the course. Section 3 describes the procedure of implementation of the modules on our campus. Section 4 summarized the evaluation of effectiveness and the consequences. Section 5 will be devoted to further considerations and future plans.

2. PROGRAM MODULES

Program modules listed below are initially developed for Apple II DOS 3.3 by us on the campus and are later transported to a TRS-80 color computer.

Utilizing adequate high resolution color graphics capabilities of microcomputers listed above, the following four modules are developed, written, and tested.

I. Particle in a box.
II. Harmonic Oscillator
III. Hydrogen Atom.
IV. Scattering (one and three dimension).

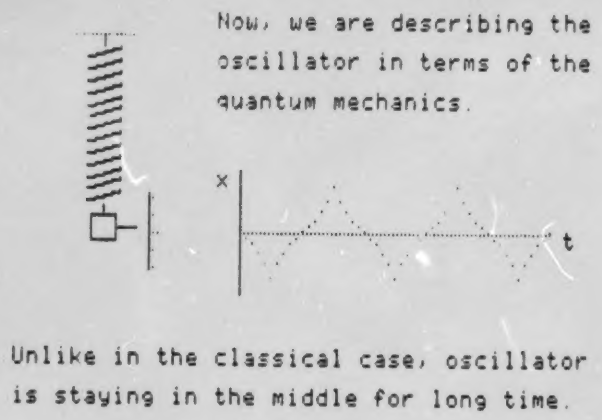
Each module contains the following submodules:

1) Description of problem comparison of classical and quantum mechanical results
2) Wave functions and orthonormality.
3) Probability density.
4) Evaluation of matrix elements.
5) Expectation values.
6) Perturbation calculations.
7) Time dependent behavior of the system.

Modules III and IV are involving three dimensional geometry, and programs are designated to provide perspective views of the display on the screen.

A high resolution color graphics, utilizing both upper and lower case leters, was used to over-lay the graphics to make the displays more attractive. Super and subscripts, scientific symbols, and other tool kits developed locally are used extensively to improve the effectiveness of the program. A few examples of screen displays as shown on the next page.

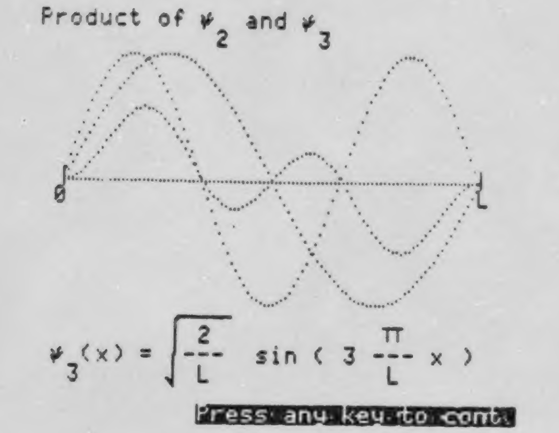
3. DEVELOPMENT AND IMPLEMENTATION



Now, we are describing the oscillator in terms of the quantum mechanics.

Unlike in the classical case, oscillator is staying in the middle for long time.

Fig. 1. Quantum mechanical discription of a harmonic oscillator

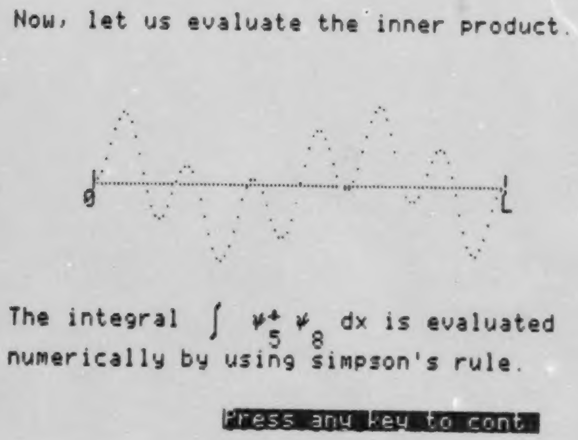


Product of ψ_2 and ψ_3

$$\psi_3(x) = \sqrt{\frac{2}{L}}\ \sin\left(3\ \frac{\pi}{L}\ x\right)$$

Press any key to cont.

Fig. 3. Wave function for a partical in a box.

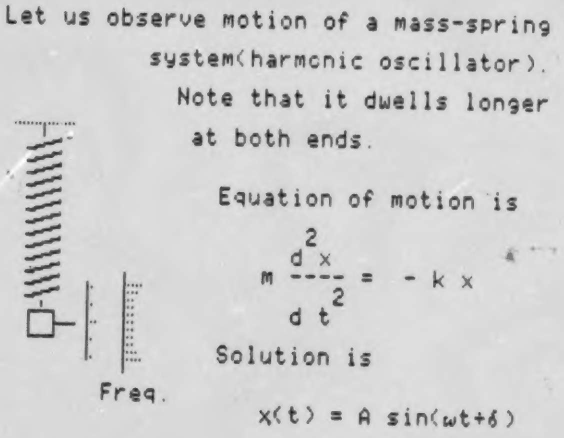


Now, let us evaluate the inner product.

The integral $\int \psi_5^* \psi_8\, dx$ is evaluated numerically by using simpson's rule.

Press any key to cont.

Fig. 5. Product of wave functions and inner product.



Let us observe motion of a mass-spring system(harmonic oscillator). Note that it dwells longer at both ends.

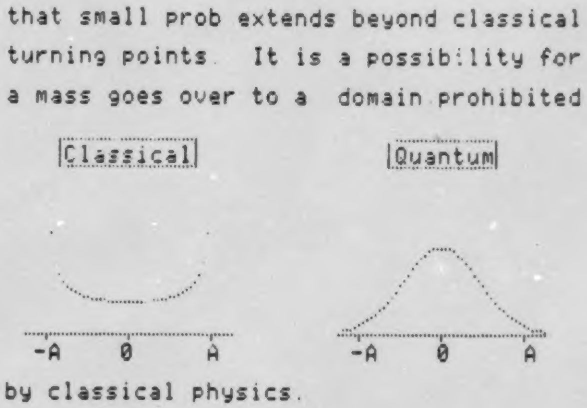
Equation of motion is

$$m\ \frac{d^2 x}{d\,t^2} = -\,k\,x$$

Solution is

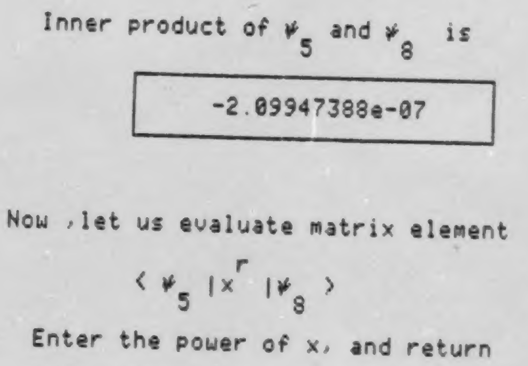
$x(t) = A\sin(\omega t+\delta)$

Fig. 2. Classical description of a harmonic oscillator.



that small prob extends beyond classical turning points. It is a possibility for a mass goes over to a domain prohibited by classical physics.

Fig. 4. Comparison of probability distributions for harmonic oscillator.



Inner product of ψ_5 and ψ_8 is

-2.09947388e-07

Now ,let us evaluate matrix element

$\langle \psi_5 | x^r | \psi_8 \rangle$

Enter the power of x, and return

Fig. 6. Evaluation of matrix elements.

The modules mentioned are developed by collaboration of the authors over an extended period without any external funding support. As the course progresses, we develop the necessary portions of the module and incorporate them into the classroom and tutorial. Students as well as colleagues provided invaluable suggestions and advice to improve the effectiveness of the program.

Wave functions are generated by using a recurrsion formula for special functions and are stored on the disk file (random access file) and were retrieved as they were needed.

The materials developed are used for 10 to 15 minutes in the classroom lecture as illustrative aids, and later on left with students for an extended period of time for the tutorial and their further examinations.

4. SUMMARY AND EVALUATION

We used these new modules for the following three courses on ETSU campuses.

1) Undergraduate Wave Mechanics (Spring, 1981).
2) Graduate Quantum Chemistry (Fall, 1981).
3) Graduate Quantum Mechanics (Spring, 1982).

Students found that it was easier to comprehend a rather abstract concept of orthogonality of wave function, probability densities, and others through these modules. Furthermore, they appreciated the descriptions of problems utilizing animation and it enhanced their comprehension of the nature of the processes involved.

Programs designed to emphasize the interacting mode were effective in keeping their keen interest and the system's displays of movement was enthusiastically received by students. Overall evaluation of the effectiveness of this technique was favorable.

5. FURTHER CONSIDERATIONS

It was found that microcomputers with high resolution graphics capability with only one disk drive was adequate to aid advanced science students in understanding and learning a complex subject. We plan to extend this approach to other courses of similar difficulty, e.g. mathematical physics, advanced electricity and magnetism and advanced mechanics.

SM1802 A MICROCOMPUTER SIMULATOR

Carl Schmiedekamp
Dept. of Mathematical Sciences
Drexel University
Philadelphia, PA 19104
(215) 895-2675

ABSTRACT

A microprocessor simulator and its use in a course on computer structure and organization is described. The program simulates five different microcomputers using a CDP1802 microprocessor. The use of the simulator has several advantages. Some details of the simulator which make it appropriate for the course are described.

BACKGROUND

The SM1802 simulator program was designed for a computer structure and organization course. The typical student has had one or two terms of instruction in programming with a high level language. The course does not usually include assembly language programming because that topic is taught in a following course.

We have not been satisfied with the content of the course and have decided to take an experimental viewpoint by trying different approaches which seem to promise a better course organization. For the Fall term of 1981 we decided to try using a microcomputer simulator to illustrate the concepts discussed in the class. The simulator forces the students to understand the operation of a computer in some detail. In a sense the simulator is a verification mechanism to check that the student understands the concepts involved.

The simulator program was influenced by the computer facilities available. We have a Prime 400 minicomputer which is a research tool and was available for use only for program development. The only machine available to the students was an IBM 370 on which computer time was purchased. The IBM 370 was used almost always in a batch mode and therefore the simulator was designed to operate in a batch mode only.

DESIGN GOALS AND INFLUENCES

The simulator was to be used to illustrate concepts about computer structure and the operation of a computer system. It was not designed with an emphasis on teaching assembly language programming techniques. This emphasis on computer operation led to the decision to not provide a cross-assembler for the simulated computer but rather to do all programming in hexadecimal machine language. The microprocessor simulated (CDP1802) has a simple enough instruction set that hand assembly can be accomplished without much difficulty [1].

The batch nature of usage for the simulator influenced the level of diagnostic information provided to the user. The intent of the diagnostic information is that enough data is available from a single execution to do most debugging. This is important because the insertion of any output statements into a program to examine intermediate results requires a new hand assembly of the program. It is also important that the diagonistics and error messages be understandable by novice programmers.

A secondary goal in the development of the simulator was that the simulated machines be realistic in the sense that they could actually be built using the CDP1802 microprocessor. Another secondary goal was that the program be easily transported from one system to another, in particular from the Prime minicomputer to the IBM 370.

USER LEVEL DESCRIPTION

A more complete description of the use of SM1802 is in the mini-manual in the appendix so only a brief overview will be presented here. The simulator input consists of three sections. The first section provides parameters to the simulator, the second section is the hexadecimal program to be loaded, and the third section is the data for the simulated machine to read.

The parameters for the simulator redefine such things as the time limits, the number of errors allowed, or the location of a dump. All simulator parameters have default values so that only changes need to be specified. The specified values must fall within acceptable limits or the change will be rejected as invalid. The only parameters routinely used are USERID, which identifies the person running the program, and MACHINE, which specifies which of the 5 available machines is to be simulated.

All three sections of the input data are read and listed before the execution of the user's program. The user's program and input data are both input in hexadecimal. The hexadecimal data format allows blanks between digits and comments at the end of a line.

The diagnostics produced by the simulator include a one line per instruction trace of the execution and a full dump of the register contents. The trace is turned on by default and usually provides enough information to debug a program. At the termination of a user's program, a full dump is automatically produced. This dump displays the contents of all the CPU registers, the last 10 instructions executed, the last 10 branch instructions executed, and a memory dump of all locations in memory that have received any data.

The input lines which separate the input sections must begin in column 1, but all other input data is in free format. This makes it easy to prepare the input on either a card punch or an interactive terminal. The output from the simulator has been formatted to fit on an 80 column display so that the user may debug the microcomputer programs at an interactive terminal.

MACHINE DIFFERENCES

The SM1802 program simulates five different machines and additional machine configurations can easily be added if needed. It is common for real computer systems built with microprocessors to be very different from each other even when they use the same microprocessor. The five machines simulated by the SM1802 program likewise all use the same microprocessor instruction set but few programs will run on more than one machine without change.

Each of the simulated machines corresponds to a different configuration of a computer system. The machine number specifies one of those configurations and makes it easy to specify the exact configuration desired for a particular assignment. The different machines also make clear the concept that there are many aspects of a computer system besides the set of instructions which it executes. The narrow instruction set viewpoint is common to students who see their assignments as just writing programs in a new language. The machines differ in the amount of available memory, in the method used for I/O and in whether or not the programmer must explicitly synchronize the program with the input data stream.

All machines handle output synchronization as the responsibility of the output device but some machines require special procedures to insure that a datum is ready when the input operation takes place. This feature was included to illustrate some of the problems involved with synchronizing two different processes. The three methods used by different machines are automatic synchronization, synchronization via a data ready flag, and synchronization via a CPU interrupt. Each machine is a simple machine with one method of I/O and one method of data synchronization.

The two non-automatic synchronization methods require the user's program to process the input data when it is available rather than when it is needed by the program. In the non-automatic synchronization methods, the time between input data being available and the amount of time that a particular input datum is valid varies. The students have to work with machines that may yield different results for two executions of the same program and input data. This unrepeatability is common in real systems but is usually too complicated or too subtle to analyze in an introductory course.

TYPICAL PROGRAMMING ASSIGNMENTS

In Figure 1 below is a list of typical programming assignments and the topics introduced. The assignments each illustrate an aspect of computer operation and are designed to gradually increase in difficulty. The emphasis is on the operation of the computer rather than programming techniques. For example we study arrays because they provide examples of doing explicit arithmetic on addresses which illustrates the concept that the "meaning" of the data item is dependent on its context.

PROGRAM DESCRIPTION

The SM1802 program is written in Fortran 66 because it is available on both of the systems we used. An attempt was made to make it easy to transfer the program to a different computer system. The major system dependent features are isolated in 3 of the 43 routines in the program.

The program processes each of the input sections before the user's program is executed. The input data for the user's program is read and listed and then written to a temporary file, from which the simulator reads data during the execution of the user's program. This preprocessing allows a listing of the input data along with the program, which makes it easier to check for improper data without depending on the user's program to properly list the input data. This procedure also allows errors in the input data to be listed with

Figure 1
Typical assignments and topics

1. Execute a prepared program	-using SM1802
2. Write a program to add three numbers	-programming the 1802, stacks, data pointer manipulation
3. Write a program to read a value and extract fields	-masking operations, array implementation, address arithmetic
4. Write a program to count the number of on bits in a bit field	-shifts, looping techniques, registers as counters
5. Write a program to call a subroutine which is provided	-subroutine linkage, BCD to binary conversion
6. Convert a program from machine 4 to machine 5	-asynchronous I-O, memory mapped I-O, interrupt processing

the data instead of during the execution of the user's program.

The input data device is simulated by a simple discrete simulation of a device which moves from one state to another after an amount of time. The CPU cycle count is the measure of time used by SM1802. The length of the states determines how long the data value is valid and how long the data ready signal will be on. Some randomness is introduced in the time between state changes so that the system will not be strictly reproducible.

ADVANTAGES AND DISADVANTAGES OF USING THE SIMULATOR

Several advantages were found using this simulator in our structure of the computer course. A major advantage is that the simulator provided an environment in which students can apply the concepts discussed in the classroom. For many people, the design and implementation of a program is an excellent means for them to certify that they understand a problem and its solution. The SM1802 simulator allowed the students to apply such topics as masking operations on bit patterns, different I/O methods, CPU interrupts, subroutine linkage, 2's complement arithmetic, multiprecision arithmetic, and addressing modes. The 1802 instruction set is used in the classroom as a language in which questions about the above topics may be asked and answered. The use of the simulator also provides a set of machines to which other computer systems can be compared.

The major disadvantage to using the microcomputer simulator is that extra classroom time is required to teach the mechanics of using the simulator and machine language programming. This extra time for the simulator forced us to delete or curtail the discussion of several topics that had previously been included in the course. The topics curtailed include floating point formats, microprogramming, and digital logic.

CONCLUSION

A student-oriented microcomputer simulator has been described for a computer structure and organization course. A feature of the simulator is that several different machines are simulated, each designed to highlight common computer system features. The use of the simulator was found to have several advantages when compared to teaching the course without any programming exercises. The only major disadvantage of using the simulator is that several lectures are required to explain the specific details of its use.

REFERENCES

[1] User manual for the RCA CDP1802 Microprocessor, RCA Solid State Division, Somervile, N.J.

APPENDIX

CONTENTS

I. Introduction
- A. General Operation
- B. Input Sections

II. Simulator Parameters
- Table I (Keyword Summary)

III. Machine Dependent Features
- Table II (Machine Features)
- Table III (I/O addresses)

Sample Input deck with JCL

Sample Output

1. Introduction

A. General Operation

The SM1802 program simulates the loading of a program and the execution of that program on a microcomputer. The SM1802 program simulates several machines which use a CDP1802 microprocessor as their CPU. The instruction set of each machine is the same, they only differ in the way the memory and input-output functions.

The simulator provides extensive error checking and execution monitoring capabilities to allow easier debugging of the machine language programs. The simulator can provide an instruction by instruction trace of the execution of the user's program and can provide complete register dumps at selected points in the program.

B. Input Sections

There are three sections to the data read by SM1802 each section is terminated by a dollar sign ($) in column one of the input record. Figure 1 is a sample input deck for SM1802.

The first section assigns values to the different simulator parameters. These parameters include the machine to be simulated, the address where instruction tracing should begin, the instruction locations where register dumps are to be performed, and the identification number of the user. This section uses keyword input. Keywords which are not explicitly assigned a value retain their default values. The keywords and their default values are described in section II of this manual.

The second section of the input data is the hexadecimal representation of the instructions and data to be loaded into the simulated memory of the microcomputer. The input is free-form, i.e. blanks are ignored. The hexadecimal data on a line may be followed by comments if the hexadecimal data is terminated with a semicolon (;). Each line must specify some multiple of 8 bits of data. If an odd number of hexadecimal digits appears on a line a 0 is appended to the last digit to form the last 8 bit value.

Figure 1
Sample Input Deck

```
USERID = 79123456
MACHINE = 1
$
;  --- SAMPLE PROGRAM ---
F8 00 ;         ---- SET UP X
A2    ; PLO R2 -
E2    ; SEX R2 -
;
69    ; INP 1  ---- READ VALUE
FC 05 ; ADI 05 ---- ADD 5
;
52    ; STR R2 ---- OUTPUT SUM
61    ; OUT 1  -
;
00    ; IDL    ---- END SIMULATION
$
$HEX
0A    ; ONLY ONE INPUT VALUE NEEDED
$
```

The third section of the input data is the hexadecimal data to be read by the 1802 microprocessor program running on the simulator. This data section must begin with the four characters "$HEX" in columns 1 through 4 of the first card of this section. Hexadecimal input data begins on the following card using free-form input like that described previously for the hexadecimal loader section.

II. Simulator Parameters

The simulator parameters control the operation of the simulator. The parameters are described below in alphabetical order. The limits, the default values, and the radix for these parameters are given in Table I.

DUMPA DUMPB DUMPC and DUMPD

These keywords define instruction locations at which a full register dump is to be performed. Each time an instruction beginning at a specified location is executed a dump is performed. The dump, like the instruction trace, is a description of the state of the machine after the execution of the selected instruction. The location is read in hexadecimal (base 16). The four keywords allow four different dump locations to be defined. The dump locations do not have to be defined in numerical order (e.g 0015 before 0244) or in alphabetical order (e.g. DUMPA before DUMPB).

MACHINE

This keyword specifies which set of machine dependent features are to be used. Machine 1 is the default machine if the machine is not specified. See section III below for a description of the machines available and their features.

MXCYCLES

This keyword sets the limit on the number of simulated CPU cycles that may be used to execute the user's microcomputer program. The simulation is terminated when the cycle count limit is reached. The default limit is 500 cycles and since most instructions require 2 cycles the default limit allows up to 250 instructions to be executed.

MXDUMPS

This keyword defines the maximum number of full register dumps permitted. After the dump limit is reached a single line trace is produced instead of a full register dump.

MXERRORS

This keyword sets the maximum number of errors allowed before the simulation is terminated. Each error increments the error counter and when the error count exceeds the value of MXERRORS the simulation terminates. Warning messages are not counted but user errors and internal simulator errors are both counted. The default value allows 20 errors.

TRMAX

This keyword sets the maximum number of trace outputs. After this limit is reached no more statements will produce trace output but the simulator will continue execution of the user's program. The default value for TRMAX is 1000.

TRSTART

This keyword defines the instruction location at which tracing should begin. The default location is 0, which causes all instructions to be traced until the limit set by TRMAX is reached.

USERID

This keyword defines the user's ID number. This parameter should be used to identify the user. The user number is used on several output headings produced by the SM1802 program.

III. Machine Dependent Features

There are two features which differ among the different machines simulated by the SM1802 program. The first feature is the amount of memory and how it is connected to the CPU. The second feature is the methods used for input and output.

Two kinds of memory are used. The first kind is for small machines and has only 256 8-bit words. This small memory is connected only to the lower 8 bits of the 16 bit address produced by the CPU, so that the addresses 0015 and C415 (hexadecimal) both address the same location. The use of the 8 bit address makes programming simplier because the upper 8 bits of an address need not be loaded into a register. The second kind of memory has 1024 memory locations and uses all of the address bits to specify the memory location.

There are two aspects of input-output that may differ between machines. The first aspect is whether I/O instructions are used or memory mapped I/O is used. The second aspect is the method used to synchronize the input operation. Programmed I/O does input and output using the CPU "INP 1" and "OUT 1" instructions. Memory mapped I/O does input and output by reading and writing to special memory addresses.

The input operation can be synchronized with the input data by three methods: automatic input, data ready flag, or CPU interrupts. The automatic input method reads a new value every time input is requested. In the automatic method the input data is available when requested and there is no need to consider the operation of the input device.

The data ready flag method simulates input from a device which signals the CPU when it has an item of data, after some time that item becomes unavailable. The CPU flag EF1 is used with programmed I/O to allow the user's program to test when data is available. When EF1 is 1 the data is ready and when it is 0 the data is not available. When memory-mapped I/O is used the status register is used instead of the EF1 flag.

The CPU interrupt method is similar to the data ready flag method except that the CPU's interrupt line is brought low (to 0 state) at the beginning of the time that the input data is valid. When interrupts are enabled (IE = 1) the CPU interrupt signal causes an interrupt operation to be performed when the data becomes valid. Table II and Table III describe the different features of the machines and the addresses used for the memory mapped I/O.

TABLE I

Keyword Parameter Value Summary

Keyword	Radix	Default	Minimum	Maximum
DUMPA	16	7530	0	FFFF
DUMPB	16	7530	0	FFFF
DUMPC	16	7530	0	FFFF
DUMPD	16	7530	0	FFFF
MACHINE	10	1	1	10
MXCYCLES	10	500	0	999999
MXDUMPS	10	5	0	100
MXERRORS	10	20	0	99999
TRMAX	10	1000	0	99999
TRSTART	16	0	0	FFFF
USERID	10	81123456	0	99999999

TABLE II

Machine Features

Machine #	Memory	I/O type	I/O Synchronization
1	256 words	Programmed	Automatic
2	1024 words	Programmed	Automatic
3	1024 words	Programmed	Data ready flag
4	1024 words	Memory-mapped	Data ready flag
5	1024 words	Memory-mapped	CPU interrupt

TABLE III

Memory mapped I-O Addresses

F001	Input data buffer
F002	Status register
F003	Output data buffer
F004	Control register

SAMPLE JOB DECK (with JCL)

```
//CARL JOB (######,S???,0:02),'CARL SCHMIEDEKAMP'
/*PASSWORD ####
//RUN EXEC PGM=SM1802,REGION=100K
//STEPLIB DD DSN=U.P760547.N621.SIMLIB,DISP=SHR
//FT17F001 DD SPACE=(1024,(1,3),RLSE),DCB=(RECFM=VS,LRECL=8,
//            BLKSIZE=1024),DISP=(NEW,DELETE),UNIT=VIO
//FT06F001 DD SYSOUT=A
//FT05F001 DD *
 USERID = 11111111
$
F8 00 ; LDI 00
FC 23 ; ADI 23
F8 A0 ; LDI A0
A2 ; PLO R(2)  -- SETUP I/O STACK
E2 ; SEX R(2)
69 ; INP 1
22 ; DEC R(2)
69 ; INP 1
22 ; DEC R(2)
69 ; INP 1
22 ; DEC R(2)
69 ; INP 1
22 ; DEC R(2)
69 ; INP 1
22 ; DEC R(2)
69 ; INP 1
61 ; OUT 1
61 ; OUT 1
61 ; OUT 1
61 ; OUT 1
61 ; OUT 1
61 ; OUT 1
00 ; IDL  -- END PROGRAM
$
$HEX
11 22 33 44
55 66
$
/*
//
```

SAMPLE OUTPUT

```
SM1802: A SIMULATOR FOR THE 1802 MICROPROCESSOR / V1.50 / 82- 1-11  16:24:24
 USERID = 11111111
$ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $
 F8 00 ; LDI 00
 FC 23 ; ADI 23
 F8 A0 ; LDI A0
 A2 ; PLO R(2)  -- SETUP I/O STACK
 E2 ; SEX R(2)
 69 ; INP 1
 22 ; DEC R(2)
 69 ; INP 1
 22 ; DEC R(2)
 69 ; INP 1
 22 ; DEC R(2)
 69 ; INP 1
 22 ; DEC R(2)
 69 ; INP 1
 22 ; DEC R(2)
 69 ; INP 1
 61 ; OUT 1
 61 ; OUT 1
 61 ; OUT 1
 61 ; OUT 1
 61 ; OUT 1
 61 ; OUT 1
 00 ; IDL  -- END PROGRAM


---MEMORY DUMP FROM 0000 TO 00FF, IN HEXADECIMAL
    0: F8 00 FC 23  F8 A0 A2 E2
    8: 69 22 69 22  69 22 69 22
   10: 69 22 69 61  61 61 61 61
   18: 61 00 ** **  ** ** ** **
$ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $
$HEX
 11 22 33 44
 55 66
 $

 $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $

 RUNNING MACHINE # 1

 0000/ IN:F8 P:0 D:00 DF:0 X:0 R(X):0002=>FC Q:0 INT:1 IE:1 R8:9A2E=>34 CY:     2
 0002/ IN:FC P:0 D:23 DF:0 X:0 R(X):0004=>F8 Q:0 INT:1 IE:1 RC:03EF=>34 CY:     4
 0004/ IN:F8 P:0 D:A0 DF:0 X:0 R(X):0006=>A2 Q:0 INT:1 IE:1 R8:9A2E=>34 CY:     6
 0006/ IN:A2 P:0 D:A0 DF:0 X:0 R(X):0007=>E2 Q:0 INT:1 IE:1 R2:91A0=>34 CY:     8
 0007/ IN:E2 P:0 D:A0 DF:0 X:2 R(X):91A0=>34 Q:0 INT:1 IE:1 R2:91A0=>34 CY:    10
 0008/ IN:69 P:0 D:11 DF:0 X:2 R(X):91A0=>11 Q:0 INT:1 IE:1 R9:540A=>69 CY:    12
 0009/ IN:22 P:0 D:11 DF:0 X:2 R(X):919F=>34 Q:0 INT:1 IE:1 R2:919F=>34 CY:    14
 000A/ IN:69 P:0 D:22 DF:0 X:2 R(X):919F=>22 Q:0 INT:1 IE:1 R9:540A=>69 CY:    16
 000B/ IN:22 P:0 D:22 DF:0 X:2 R(X):919E=>34 Q:0 INT:1 IE:1 R2:919E=>34 CY:    18
 000C/ IN:69 P:0 D:33 DF:0 X:2 R(X):919E=>33 Q:0 INT:1 IE:1 R9:540A=>69 CY:    20
 000D/ IN:22 P:0 D:33 DF:0 X:2 R(X):919D=>34 Q:0 INT:1 IE:1 R2:919D=>34 CY:    22
 000E/ IN:69 P:0 D:44 DF:0 X:2 R(X):919D=>44 Q:0 INT:1 IE:1 R9:540A=>69 CY:    24
 000F/ IN:22 P:0 D:44 DF:0 X:2 R(X):919C=>34 Q:0 INT:1 IE:1 R2:919C=>34 CY:    26
 0010/ IN:69 P:0 D:55 DF:0 X:2 R(X):919C=>55 Q:0 INT:1 IE:1 R9:540A=>69 CY:    28
 0011/ IN:22 P:0 D:55 DF:0 X:2 R(X):919B=>34 Q:0 INT:1 IE:1 R2:919B=>34 CY:    30
 0012/ IN:69 P:0 D:66 DF:0 X:2 R(X):919B=>66 Q:0 INT:1 IE:1 R9:540A=>69 CY:    32
 0013/ IN:61 P:0 D:66 DF:0 X:2 R(X):919C=>55 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:    34
 0014/ IN:61 P:0 D:66 DF:0 X:2 R(X):919D=>44 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:    36
 0015/ IN:61 P:0 D:66 DF:0 X:2 R(X):919E=>33 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:    38
 0016/ IN:61 P:0 D:66 DF:0 X:2 R(X):919F=>22 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:    40
  HEXOUT(    1): 66 55 44 33 22
```

```
0017/ IN:61 P:0 D:66 DF:0 X:2 R(X):91A0=>11 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:   42
0018/ IN:61 P:0 D:66 DF:0 X:2 R(X):91A1=>34 Q:0 INT:1 IE:1 R1:DD8E=>34 CY:   44
0019/ IN:00 P:0 D:66 DF:0 X:2 R(X):91A1=>34 Q:0 INT:1 IE:1 R0:001A=>34 CY:   46
PROGRAM STOPPED (IDL) AT LOCATION 0019
 HEXOUT(   6): 11

       --TERMINATION DUMP--
MACHINE: 1/CYCLES:    46/ERRORS:   0  /   11111111      64
    LOC:0019    P:0       D:66   EF1:0     Q:0
   NEXT:001A    X:2      DF:0    EF2:0   INT:1
     IN:00      T:AD     IE:1    EF3:0   EF4:0

  R(0):001A  M(R(0)):34    R(8):9A2E  M(R(8)):34
  R(1):DD8E  M(R(1)):34    R(9):540A  M(R(9)):69
  R(2):91A1  M(R(2)):34    R(A):2A60  M(R(A)):34
  R(3):40F5  M(R(3)):34    R(B):5F22  M(R(B)):34
  R(4):1519  M(R(4)):00    R(C):03EF  M(R(C)):34
  R(5):3FA6  M(R(5)):34    R(D):D21E  M(R(D)):34
  R(6):8033  M(R(6)):34    R(E):FB33  M(R(E)):34
  R(7):83FD  M(R(7)):34    R(F):D5E5  M(R(F)):34

 -- LAST 10 INSTRUCTIONS
     LOC:0010  IN:69
     LOC:0011  IN:22
     LOC:0012  IN:69
     LOC:0013  IN:61
     LOC:0014  IN:61
     LOC:0015  IN:61
     LOC:0016  IN:61
     LOC:0017  IN:61
     LOC:0018  IN:61
     LOC:0019  IN:00

 -- LAST  0 BRANCH INSTRUCTIONS

   DUMP-COUNT:  1    TRACE-COUNT:   23


 ---MEMORY DUMP FROM 0000 TO 00FF, IN HEXADECIMAL
    0: F8 00 FC 23  F8 A0 A2 E2
    8: 69 22 69 22  69 22 69 22
   10: 69 22 69 61  61 61 61 61
   18: 61 00 ** **  ** ** ** **
   98: ** ** ** 66  55 44 33 22
   A0: 11 ** ** **  ** ** ** **
```

INTERFACING WITH ATARI 800 PADDLE PORTS

Kenneth E. Williams
Michael F. McInerney*

Microcomputers in Science Education and Research Group
and Physics Department

Eastern Illinois University
Charleston, Illinois
217 581-3420

INTRODUCTION

In this paper we describe a simple method for using the Atari 800 paddle ports as analog interfaces. The first section of the paper is concerned with the necessary hardware while the second describes a software routine used to sample the ports. This routine also addresses the crucial problem of timing and resolves it by using the Atari's internal clock.

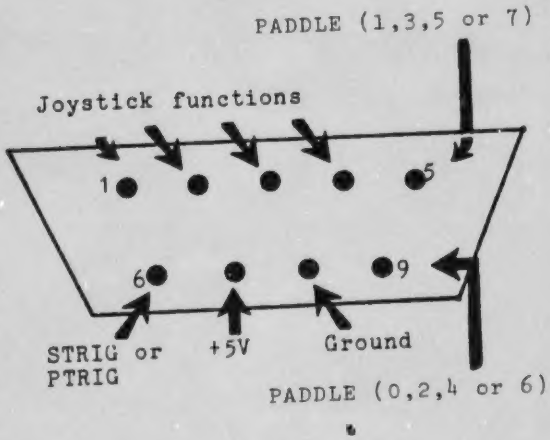


Fig. 1 ATARI paddle-port pin map[1].

HARDWARE

The Atari paddle and joystick port has nine pins. Numbering from the top left, as in Figure 1, pins 1-4 respond to the joystick, pins 5 and 9 correspond to two paddle inputs, and pins 7 and 8 are the +5V and ground, respectively. You will notice that the +5V and ground are very close to one another; presumably it would be bad for the machine if they were inadvertently connected. Such things do occasionally occur, however, but we are happy to report that the worst that has happened is loss of the current program.

The Atari paddle control itself is a simple potentiometer which is connected across the +5V and one of the paddle inputs. A range of the potentiometer between 0 and about 1 M Ω produces a variation of between 0 and 228 in the value of the Atari-Basic PADDLE function.

The computer 'reads' the potentiometer resistance by incorporating it into a simple R-C circuit as shown in Figure 2. The timer, located in the POKEY chip, oscillates with a frequency dependent upon the resistance; this frequency is stored by the PADDLE function. As far as the simple interfacer is concerned, the POKEY chip discharges the capacitor every cycle, which is the most important feature. This varying voltage may cause difficulties if anything other than a resistance is placed in the circuit.

Thus, it is relatively simple to interface a large resistance, or anything that may be made to resemble a large resistance, to the Atari through its game paddle ports. The authors have only recently begun to explore this facility, and three of our simple experiments are described below.

---

*Dr. McInerney will deliver the talk.

The Pendulum

The circuit for this experiment is shown in Figure 2. One side of the potentiometer, resistance R in the figure, was connected to the +5V (pin 7) while the tap was connected to the paddle pin (pin 5). Oscillations of the pendulum were converted directly into resistance variations by using the potentiometer control as the pendulum fulcrum. Although simple and effective, this arrangement necessitated the use of a heavy rigid pendulum (a meter of 1 inch by 1/8 inch angle iron) to overcome friction within the potentiometer.

The Photocell

A cadmium sulfide photocell has a resistance variation in the range 10Ω to 1 M Ω, corresponding to high and low light intensity. We connected the photocell across pins 5 and 7, as with the pendulum, and were immediately able to read the light intensity in a scale 0 to 228. 0 corresponds to no resistance and hence high light intensity.

Although the photocell has good sensitivity, it is relatively slow to respond and thus, cannot be used in timing experiments. Its most obvious application is in chemistry as a spectrum analyzer.

The Light-Sensitive Transistor

The FPT 100 silicon phototransistor has an emitter-collector current dependent on the intensity of light -- the higher the intensity, the higher the current. The transistor also has a very rapid response time. Initially, we tried connecting the emitter and collector across pins 5 and 7 in the same manner as the potentiometer. Unfortunately, the action of the chip inside the computer interfered with the transistor; fluctuations occurred in the buildup of charge on the capacitor which resulted in large uncertainties in the reading.

The simplest way to resolve this problem, or rather to reduce the uncertainty of the reading to a few percent, was to use a 3μF capacitor in parallel with the resistance R as shown in Figure 2. In this configuration of high light intensity corresponds to a reading of 228 in contrast to the photocell.

The experiments considered so far have all used large resistances or effective resistances. In the near future we hope to turn small voltage and current effects into large apparent resistors using circuits such as that of Figure 3. In this way we will be able to use strain gauges, variable transformers, and heat resistant diodes.

SOFTWARE

The Atari 800 contains a resident real-time clock which can be used to determine the time interval between samplings of data from one or several paddle ports. The data can then be recorded in arrays along with the corresponding time of sampling by means of a software routine. Such a routine enables storing of the data on a disk or cassette from which it can be retrieved for future processing.

The real-time clock (RTCLOK) resides at decimal memory locations 18-20 in RAM. Location 20 contains the low-order byte of the timer, location 19 the middle-order byte, and location 18 the high-order byte. The RTCLOK is initialized when the machine is powered up or the system is reset. The low-order byte is actually a counter which increments itself every 16 msec and wraps around every 4.096 sec according to a preset machine cycle. The middle-order and high-order bytes serve as overflow counters, allowing for a total time of over 74 hours before the timer wraps around to zero. The timer, therefore, allows for an impressive range of data-sampling times. For most purposes, though, the high-order byte need not be read.

We devised a Basic routine called GRAB which samples up to 100 sets of data from three paddle ports, each taken simultaneously, and correlates these with RTCLOK (see Acknowledgment). A listing of this routine is provided in Figure 4.

The program asks for two parameters initially: (1) the number of data samplings to be taken, and (2) the number of delay loops required to provide an appropriate frequency of data sampling. The latter cannot be correlated precisely with a given time interval since the Atari Basic interpreter produces an object program of varying length in RAM. This variance, however, is not too great and can be tolerated for all applications where precise time intervals are not critical to the experiment.

The first part of the program permits the determination of an average time between samplings, corresponding to the number of delay loops initially specified. The program runs through the actual sampling loop a number of times, also as initially specified. The RTCLOK is read for each sampling, time intervals between samplings computed, and an average time interval calculated and displayed. If the time interval is inappropriate, the program allows for returning to the first step in order to input new parameters.

The lower limit of timing for our particular routine is around 0.07 sec per sampling as the array corresponding to one delay loop. If faster timing is required, unnecessary steps can be removed from the sampling loop. For example, if only one paddle port must be used, statements 110, 115, 180, and 190 in our program could be deleted, thereby reducing the total execution time in RAM.

There is a potential problem associated with the paddle ports. Since a finite interval is required for translation and execution of each Basic instruction, it is impossible to read information from more than one port simultaneously. We have provided a partial solution to this dilemma, however. The program reads information from two of the three ports twice in a particular sequence and averages the values read. In this way we interpolate between readings so that the information recorded corresponds to the data available from all ports at about the same point in time.

Once the appropriate parameters are determined, the program can then be used to perform the actual sampling of data. Paddle port information and times are stored in arrays. The arrays can then be stored on some peripheral device for future reference. In our own case the data are read into a file on disk called GRABFIL with corresponding elements of each array stored in sequence. Different storage schemes, including storage on cassette or tape, are also possible, of course.

CONCLUSION

We have found analog interfacing with the Atari 800 to be relatively simple. The few applications we have made, however, are only crude pointers toward what could be done. In particular we have as yet made no attempt to use more than one of the paddle ports or to realize the full potential of the three microprocessors in the Atari CPU. We feel that the possibilities are many and varied. Air-track demonstrations, cooling-curve experiments, and damped and multiple pendulums, among others, immediately come to mind as useful applications.

ACKNOWLEDGMENT

We should like to acknowledge our debt to Dr. R. Tinker of TERC, Cambridge, Massachusetts. It was from his Chautauqua Short Course "Microcomputers in the Laboratory" that we received the impetus and many of the ideas for this paper.

REFERENCES

[1] *ATARI Personal Computer System Operating System User's Manual and Hardware Manual* (ATARI, Inc.: November 1980), Appendix C.

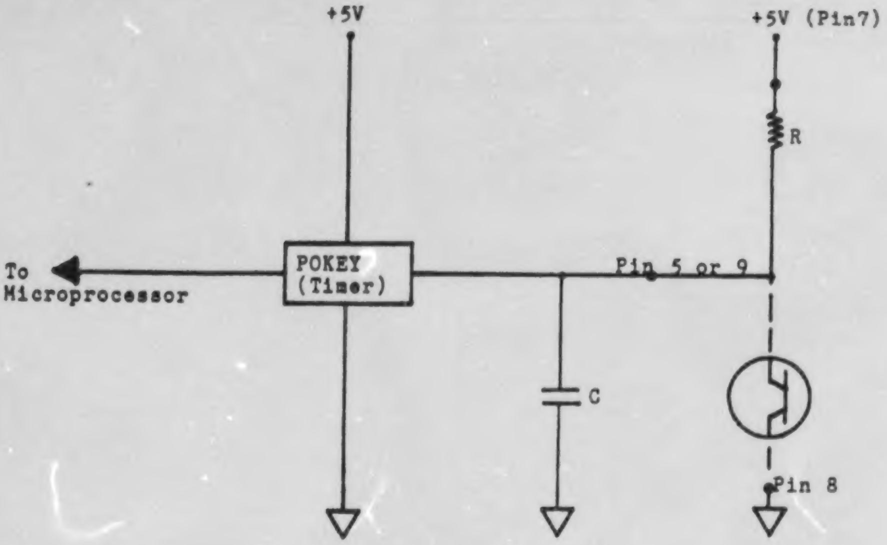

+5V
+5V (Pin7)
R
To Microprocessor
POKEY (Timer)
Pin 5 or 9
C
Pin 8

Fig.2 Schematic diagram of potentiometer and photo-transistor arrangements. In the former case, the transistor circuit (dashed line) is omitted and the resistance R is variable. In the latter case, the transistor is included and the resistance R is kept fixed.

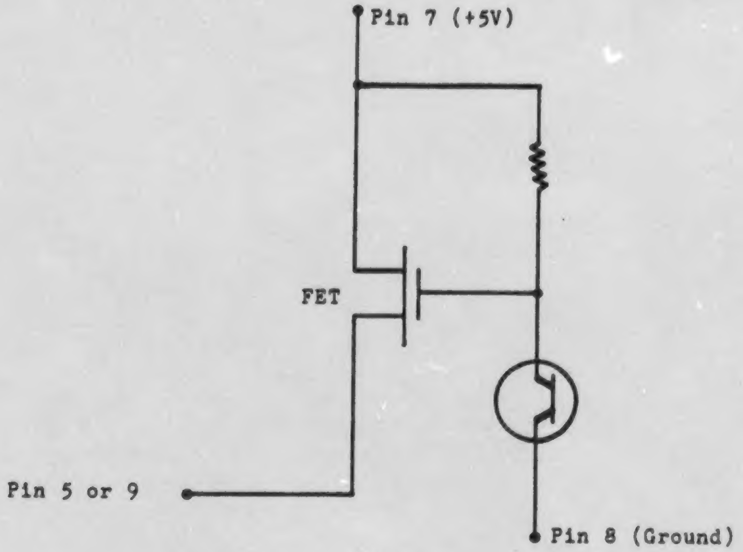

Pin 7 (+5V)
FET
Pin 5 or 9
Pin 8 (Ground)

Fig. 3. Schematic diagram showing an arrangement using an FET, in which a transducer (in this case, the photo-transistor) can be interpreted as a resistance.

Special Session

INSTRUCTIONAL MIS: USER AND MANAGEMENT CONSIDERATIONS

Chaired by: Robin D. Avery
Macomb Community College
Wayne State University

ABSTRACT

This session will address the complexity of and provide a rational for instructional MIS development. The major considerations are: time, personnel, hardware, software, training, and other needed resources (includeing associate costs).

MIS can be a useful strategy for some, but inappropriate for other instructional programs. For those individuals/institutions interested in MIS development, the advantages and limitations must be presented. Also examples of "successful" automated applications will be shown for comparison with similar activities in a manual environment. Thus, in order to illustrate the information processing capability of MIS, instructional objectives, computerized applications, and data manipulations (outputs) will be shown from the distinct disciplines in higher education: English Composition and Cooperative Education.

Examples of programming will be included from both areas: Co-op Management Information Systems (CMIS) student reporting (FORTRAN); and English Composition: Essay Grading (FORTRAN), and Computer Managed Instruction (CMI), (FORTRAN, BASIC, TEXT EDITOR, SPSS). All relevant documentation will be available and shared. Details for contacting colleges and universities that have developed instructional MIS (175 est) will also be provided. In addition, research implications will be discussed in relation to the output examples of instructional data manipulation, especially in the area of English composition.

Special Session

TEACHER TRAINING PROGRAMS: USE OF MICROCOMPUTER

Chaired by: Forrest L. Coltharp
Pittsburg State University

ABSTRACT

Various approaches to training teachers in the use of microcomputers in their classroom are presented through the programs discussed by this panel.

Participants

Elton Beougher
Ft. Hays State University

Mary K. Corbitt
Kansas University

Richard T. Driver
Topeka Public Schools

John J. Rezac
Shawnee Mission Schools

Project Presentation

APPLICATIONS IN THE UNDERGRADUATE SCIENCE CURRICULUM

Chaired by: Jean Rogers
University of Oregon

Mark Adams
Donald Collins
Victoria Collins
Todd Clonniger
Richard Cornelius
John M. Franz
Helen H. Neufeld
Teron Rockhill
James D. Spain

ABSTRACT

Computerized Data Logging

Mark Adams & Todd Clonninger, Students; Donald Cllins, Faculty in Physics; Victoria Collins, Faculty in Chemistry, Warren Wilson College, Swannanoa, North Carolina 28778

The purpose of the project was to make computerized data logging available to students in all levels of chemistry and physics classes especially for introductory levels. This is particularly useful to small colleges with small budgets and a desire to introduce students at elementary levels to the computer.

There are two versions of the program-one in Pascal, the other in Basic and Machine Language. Both require some programming by the user to retrieve and analyze the data once it has been collected. In general, the programs read an analog to digital conversion device, store the values to a disk, and display a graph of the data. Any equipment that produces an analog voltage output can be used with the proper voltage conditioning. The Pascal version has a variable time interval and a maximum of 512 readings. The Basic version has a fixed sampling rate of one per second and up to 16,656 readings.

The additional programming necessary to make the programs specific enough for a particular experiment is not great. The extra equipment necessary is not expensive. The benefits in minimizing the drudgery of manual readings are impressive and they can be used by almost anyone.

The results, successes and problems of introducing computer data logging at the introductory levels will be presented.

ABSTRACT

Microcomputers and Large Screen Projection in Science Lecture Halls

Richard Cornelius, Chemistry Department, Wichita State University, Wichita, Kansas 67208

Microcomputers (and NSF) are supporting new avenues for teaching chemistry at Wichita State University. The combination of color graphics with speed and accuracy of computer calculations supports a different approach to classroom lectures. Apple microcomputers have been linked to large screen video projectors in order to display computer-generated images to large classes. The emphasis in this approach has been "interactive" with the goal of a give-and-take between the students and the microcomputer moderated by the instructor. This dynamic technique permits manimpulations of variables and allows experimental exploration to questions of "What would happen if...?". Computer programs written for use in this mode rely heavily upon visual and graphical images, use color and sound, and are organized to emphasize rapid system response to changes in variables while masking the complexity of the calculations. A demonstration with the projection equipment as used in the classroom will utilize programs from chemistry, physics, engineering, mathematics, and computer science as time permits.

ABSTRACT

The Molecules in Our Body -- A Hard-Core Biochemistry Microcomputer Assisted Instruction Molecule

John M. Franz, Department of Biochemistry, 322 Chemistry Building, University of Missouri-Columbia, Columbia, MO 65211

Utilization of microcomputer-assisted

instruction as an educational tool for teaching the central dogma of biochemistry has been comparatively slow. One major reason is the paucity of software dealing with biochemical structures and metabolic pathways. The development of such software requires a microcomputer system with user-d-signable high resolution graphics and sufficient memory to contain a tutorial type module. The appearance of the low-priced 32K RAM Exidy Sorcerer (1), with a graphics resolution of 512 X 240 pixels and 128 graphic characters, each of which could be designed by the user employing an 8 X 8 dot matrix, stimulated us to develop a tutorial biochemistry module.

From a written survey of the most heterogeneous lower level biochemistry class and discussions with its instructor it was determined that the students experienced difficulty in bridging the gap between the simple organic chemical structures they studied previously and new biochemical structures. An examination of the tests, lecture notes and texts for the targeted biochemistry course and its prerequisite organic chemistry course revealed that a useful transition between structures could be made employing the functional group approach. The objective for 'The Molecules in Your Body - Part 1' (dealing with compounds which do not contain nitrogen) states: The student will be able to discuss the manner in which simple organic compounds (alcohols, aldehydes, ketones, carboxylic acids, esters and acid anhydrides) are related to important types of biochemical compunds (carbohydrates, fatty acids, fats, ADP, ATP and other metabolites). Dr. Charles A. Huibregtse wrote the main dialog. Mr. Gregory R. Blythe did the programming including most of the graphic design. I wrote the queries, remedials and transitions, designed the screens and coordinated the project. The completed module (Copyright 1980 - The Curators of the University of Missouri) is approximately 64K (30 minutes) long and thus requires one intermediate reload if a 32K RAM Sorcerer is employed. This program, designed for non-majors, does not include discussions of related topics such as isomerism or dissociation constants. A semi-humorous conversational tone and the liberal use of graphics are employed to maintain interest.

ABSTRACT

Barriers to the Use of Instructional Computing in the Teaching of Undergraduate Chemistry

Helen H. Neufeld, PhD, Glendale Community College, 1500 N. Verdugo, Glendale, Calif. 91208

A review of the recent literature of instructional computing shows: (a) a growing incorporation of computing techniques at the graduate level of chemical education; (b) that instructional computing is not widely used in undergraduate chemistry despite the need for computer proficiency in future chemists.

This research was undertaken to examine the hypothesis that characteristics of a collegeor university such as institutional type, size and average expenditure per student determine the successful use of instructional computing in chemistry.

Questionnaires covering many aspects of the use of computing were went to chemistry department chairmen at all accrediated colleges and universities in the southern half of California. On the basis of data from the sixty-nine replies, twelve institutions with an exemplary use of computing in chemistry were selected for campus visits and interviews. Institutions visited were of varied size and affluence, and were selected to include representative private and public colleges and universities. Chemistry department chairmen, professor, computer center directors, a university computer programmer, and a university vice president-chemist were interviewed. The forth-three persons interviewed provided much information about barriers encountered when computing was first incorporated into their chemistry instruction--barriers which could be categorized as administrative or economic barriers, faculty barriers and commercial computer supplier barriers. Their strategies for coping with these obstacles were described as well.

An analysis of data from the questionnaires and interviews showed no relationship between the institutional characteristics studied and the effective use of instructional computing in chemistry.

A single factor, the presence of one or more faculty entrepreneurs, appeared in each of the twelve institutions to determine whether computing was used effectively. In every institution one or more persons had invested a great deal of personal effort to achieve the incorporation of computing, usually with much work after hours.

These findings have implications for administrators and computer suppliers: these persons should be identified and actively supported if the full potential of computing as an instructional tool is to be realized. Financial assistance is needed from both private enterprise and the Federal government for such efforts.

ABSTRACT

SUNY Brockport Science and Mathematics Microcomputer Project

Theron D. Rockhill, Professor Math/Computer Science Department, SUNY College at Brockport, Brockport, NY 14420

The SUNY Brockport Science and Mathematics Microcomputer Project is developing and implementing instructional uses of microcomputers in undergraduate science and mathematics courses. With support from the National Science Foundation CAUSE Program, the Project has conducted faculty workshops on instructional uses of computers, instructional development techniques and computer programming; supported faculty-student (author-programmer) teams to develop instructional modules for using microcomputers in teaching science and mathematics; and implemented a wide range of instructional uses of microcomputers, including drill and practice, tutorial, problem solving, modeling and experiment simulation.

Instructional modules have been developed for a wide range of undergraduate science courses. Psychologists have written programs for collecting and analyzing data for experiments on visual illusions and memory recall. Simulation programs entitled Quantum Wells, Elastic Collisions and Wheatstone Bridge are being used by physics classes. Drill programs have been prepared for a number of topics in Biology, Chemistry, Geography, and Physics. Earth Science faculty have developed modules on solar radiation, ocean properties and faulting. In Biology, faculty have prepared modules on lab simulation for Anatomy and Physiology (a turtle heart simulation) and Genetics (natural selection).

Instructional modules have been developed for several mathematics courses, including college algebra, finite mathematics, calculus, and differential equations. Tutorial modules on matrices, linear equations, and solving linear programming problems have been prepared for a finite mathematics course. Tutorials have also been developed on the definition of derivative and on basic properties of functions. A number of function graphing programs are available for use throughout the mathematics curriculum. In addition to several statistical routines for data analysis, the project has prepared tutorials on central tendency, probability, z-scores and t-test for use with introductory statistics courses.

The development team for each module, or group of modules, usually included one or two faculty authors and an undergraduate student programmer. Each module includes a documented BASIC program written for the APPLE II Plus (48K, DOS 3.2) microcomputer and a written user's guide. Microcomputers are located in the laboratory areas of each department where they are readily available for use in labs and classrooms. Module users provide feedback on the module content and on their experience using microcomputers.

ABSTRACT

The SUMIT Development Project

James D. Spain, Department of Biological Sciences, Michigan Technological University, Houghton, Michigan 49931

The National Science Foundation, Development in Science Education Program is supporting a two year project to develop and evaluate microcomputer-based courseware for biological science instruction at the post-secondary level. The project names is an acronym for Single-concept User-adaptable Microcomputer-based Instructional Technique and thus emphasizes development of a large number of relatively small programs which are clearly written in BASIC and documented fully to permit easy adaptation to local needs and interests. It is believed that the user-adaptable approach will lead to much greater program utilization than has been experienced with previous instructional software.

The SUMIT project has just completed its second year of operation and has resulted in about 20 biology courseware modules. Each module has been evaluated in either general biology, animal ecology or cell biolgoy courses at Michigan Tech. Most evaluations involved both pre- and post-testing to determine the knowledge gained and individual student evaluations to ascertain student attitudes toward computerized instruction. In three cases, modules were compared with either lecture or text format using randomly split classes.

After these summative evaluations were completed, the modules were turned over to CONDUIT to be examined by their reriewing teams. It is expected that some of these modules will be available for review by conference attendees at the CONDUIT display. The complete package of SUMIT modules should be available from CONDUIT in late 1982 or early 1983. Some sample programs will be demonstrated at the project presentation session.

Special Session

ELECTRONIC MAIL AND COMPUTER BASED CONFERENCING

Presented by: Paul S. Heller
EDUNET/EDUCOM

ABSTRACT

Electronic Mail

Introduction to electronic mail as a medium for person to person group communication. Descriptions and illustration of basic and advanced features of several widely used mail systems. Examples of applications of mail systems in educational and nonprofit settings. Advantages and disadvantages of electronic mail in comparison to telephone and postal letter media. Description of Mailnet, a project now underway to provide low-cost automatic exchange of electronic mail among many campus based electronic mail systems throughout the country.

Computer Conferencing

Presented by John T. Hogan, Carolina Educational Computing Service

Since the use of microcomputers is essentially distributed and not centralized, electronic conferencing can help to solve the problem of how to provide a microcomputer support to an educational network. In the past year NECES has begun offering a UNIX-based electronic conferencing and mail service on a DEC LSI 11/23 for the exchange of information among microcomputer users throughout North Carolina. One significant recent development has been the expansion of the bulletin board service to the traditional users of certralized computing services. This expansion may well increase communication between these traditional and those who rely on mocrocomputers.

Computer Science

COMPUTER LITERACY IN THE ELEMENTARY SCHOOL
A DISTRICT-WIDE PROGRAM

by Mrs. Bobby Goodson
President, Computer-Using Educators
Computer Resource Teacher
Cupertino Union School District
10301 Vista Drive
Cupertino, California 95014
408-252-3000

The Governor of California recently stated that we should be working toward "technological literacy" as one of our goals in education. To that end he suggested that we should have as a first priority an increased commitment to mathematics, science, and computer instruction in our K-12 schools. He then added the goal of increased support to the universities and vocational schools in the same curriculum areas. Our experience in the Cupertino Union School District during the past four years suggests that an additional goal - perhaps a first goal - is needed if we are indeed to reach the level of technological literacy that will be demanded by the "information society" in which our students will be living. We must ensure that every elementary student is "computer literate" before entering high school!

Cupertino has made this the goal of our district-wide computer literacy program. Our philosophy is that EVERY child, before reaching the eighth grade, should have the opportunity to reach a level of computer awareness that will enable them to make good academic and career choices during the coming years. To achieve this goal we have developed a K-8 computer literacy curriculum as a guide for our teachers and have initiated an inservice program to provide training to our teaching staff. Equally important is the training we have provided for our administrators, office personnel, and parent aides.

COMPUTER LITERACY - A BEGINNING

The original Computer Literacy Program was developed under the guidance of Associate Superintendent William Zachmeier as a 1978 ESEA Title IV-C project in the junior high school. This project, funded under Title IV-C for two years, provided a full time director who developed the set of objectives which set a firm foundation for the present program.

This description of our basic project is designed to share a little of what we have learned during the four years of the project and to indicate where we hope to go in the future. It is our hope that our experience will be of benefit to those just stepping out into this very exciting new area in education.

During the developmental years we had to first define computer literacy. This widely used term, introduced by Arthur Luehrmann several years ago, has never had a universally accepted definition. We established that, for Cupertino, there are two levels of computer literacy. First, our students should become aware of computers.

At this level they learn what a computer is and more importantly what it is not. They begin to think about the effects computers are having, good and bad, on the world around them. They develop an historical perspective and place the computer in its proper position as the latest step in humanity's continuing search for better and faster means of dealing with information of all kinds. Students also acquire a usable computer vocabulary - a consumer's vocabulary if you will - that enables them to read or listen to the news and to discuss computer related subjects with understanding. They become aware of the growing career opportunities that will be open to them in this new information society. We feel that everyone needs this level of computer awareness.

The second level of computer literacy is that of computer use. Everyone needs to be able to deal with a computer as a tool - in the bank, the store, the car, the kitchen, and the workplace. This may mean only being able to use a keyboard or number pad. More importantly it means overcoming the tremendous psychological barrier some people feel toward this type of technology. Many students progress beyond this introductory level of computer use and learn to use the computer as a problem solving tool. They have the ability to use software designed by others to accomplish desired tasks. Other students go even beyond this level to investigate the logic of the computer and enter the world of computer programming. These two levels, computer awareness and computer use are combined to form our definition of computer literacy. We feel that all K-6 students should become aware of computers. All junior high students should have the opportunity to learn to use a computer as a tool and be introduced to the idea of computer programming. They will then be able to make better academic choices during their high school years.

Since the computer now impacts all areas of our day-to-day life we determined that it should not be placed in any one specific academic subject area. It has applications in every field. Our students learn about the history and development of calculating devices as well as the effects of computers on society within their regular social studies classes. They become familiar with computer vocabulary as part of their vocabulary and spelling practice in language arts classes. Mathematics is the subject area in which to introduce the concepts of logic and problem solving. It not only fits into the regular curriculum here, but also gives us a way to teach many of these ideas more successfully than ever before. Science is the only area in which we have suggested adding a specific unit on computers, probably at the junior high level. Here we introduce the computer itself, the more specific vocabulary, and some of the basic ideas of computer operation.

We have tried to introduce the computer as a new tool - a creative, innovative teaching tool for teachers and an exploratory problem solving and learning tool for students - in ALL subject areas. At the awareness level in the early grades, the presence of a computer in the school is desirable although not essential. The fact that the school does not have a computer yet, or perhaps has only one, cannot be used as an excuse for not introducing computer awareness! If the computer is looked upon as a creative, multi-purpose tool, there is much that can be done with a single computer in a school. The excuse that "we only have one computer, so what can we do?" is simply not valid!

TEACHER INSERVICE - A NECESSARY INGREDIENT

To reach our goals and to nurture this program, Cupertino had to develop a strong inservice program for our teachers. Since there is apt to be great resistance to computers, this program had to be non-threatening, non-technical, gradual - and fun! That's a tall order, but within an existing active district-wide inservice program it has become possible. The courses are designed in small modules - the introductory awareness classes may be taken in three segments of two two-hour classes each. These classes cover the awareness goals outlined above for our students, as well as an introduction to the computer as a classroom tool. Teachers are given an opportunity to see a variety of relevant software and to begin to develop a sense of what is good in educational software. This series of classes may be followed by an introduction to Basic, if teachers wish, although we stress that no teacher needs to become a programmer in order to use a computer! There are those who want to develop this level of understanding, however, and for them we offer a second 15-hour course of programming. We also offer a programming laboratory for those who simply want more time on the computer to develop their understanding. Courses in the creation of classroom material and using the various tools and authoring languages are available as well as classes in the design of educational software enabling a teacher to translate an idea into a form that a professional programmer could interpret. This year we have introduced workshops in using the computer as a word processor and as an administrative tool (using existing general purpose software). We will introduce Pilot as an alternative computer language for children and are looking forward to introducing LOGO as soon as it is available.

A LOOK TO THE FUTURE

What have we learned after four years of this comprehensive program? First and foremost, we have learned that an introduction to computers is important for ALL students - kindergarten to eighth grade, special education students to gifted. There is no apparent way to predict which student is going to be the most excited by these new ideas, and it is very often the quiet average student who has never really been touched by the excitement of learning! Students with learning problems often make great progress with this new "friend" that is ever patient and non-judgmental. Gifted students, and some who have never realized their gift for logical, creative thought, take off as never before with this marvelous open-ended, exploratory device!

We have learned that it takes time. These ideas cannot be forced suddenly upon all teachers. Computer laboratories cannot be introduced into a school without preparation and a great deal of thought and planning. Teacher training, often offered nowhere else, must be readily available within the district in easy increments and on a continuing basis. Administrators must recognize the amount of time and the dedication that teachers are giving to this new field. This type of program cannot achieve its potential without the guidance of a full-time director, who can devote full time to learning what is available and what is possible in this new field. Cupertino's original project was written in the same year that microcomputers were introduced to the general public. We as a district and I as the author and director have been able to grow and to learn as the concept of microcomputers in education has developed.

We have learned that there is no RIGHT microcomputer for all educational uses. Hardware selection depends upon the projected program. At the awareness level, almost any microcomputer for which there is adequate software to meet planned uses will work. However, we also have found that programs may quickly grow and blossom beyond the most ambitious plans; a microcomputer to meet these future needs should be versatile and easily expandable. We have found, much to our surprise, that at the awareness level there may be a definite advantage in having more than one brand of computer in the laboratory. In addition to the fact that a greater variety of software will be available and a wider range of capabilities may be utilized there is one other major advantage. Students who learn, at this early exploratory level of computer awareness, on a variety of microcomputers seem to cope easily with diverse systems and, in the long run, will be able to use whatever system becomes available to them. However, in a group classroom or laboratory where instruction is being given, as in our junior high elective classes, hardware uniformity within one site is essential. We now are using three different microcomputer systems within our district and have not found the diversity to be a problem, either in software acquisition or in maintenance.

Where are we going in the future? Under the direction of Superintendent Carl W. Krause and a very supportive Board of Education, we already have placed computer laboratories in each of our junior high schools. All junior high students have an opportunity to take a computer elective course. Ninety percent of our K-6 schools have at least one computer and five of them will have small laboratories in operation by the fall of 1982. We also plan to add labs to several more schools each year, with the goal that all schools will have labs by 1984. Sixty-five percent of our teachers have already had training in computer use. We will continue to expand their opportunities and encourage their explorations in this area. We will also explore new uses for the computer in the classroom and will work actively to improve the quality and the availability of educational software.

COMPUTER LITERACY 1982: AN UPDATE

Grace C. Hertlein, Professor
Department of Computer Science
California State Univ., Chico
Chico, California 95929
(916)895-6443

PURPOSE

One of the most popular and important courses in today's undergraduate curriculum is Computer Literacy. Its importance grows with each passing year, as more institutions require such a course for all undergraduate students. In many institutions, this need is partially met by allowing General Education credit for Computer Literacy. As the trend continues its present thrust, in the very near future, Computer Literacy will be an accepted "universal" requirement for every undergraduate who hopes to get a Bachelor's Degree. We need to plan for that universal undergraduate requirement now.

The purpose of this paper is to very briefly review the history of this type of course -- but primarily to focus on some of the newest, universal trends in this important planning, to present updated in-depth objectives for such a course, and to discuss some class-tested ideas that work.

HISTORY

This course and its many variations has been reviewed in depth during the past dozen years by many fine teachers and writers. A survey of the nine CCUC Proceedings and the three NECC volumes gives evidence of creative, changing approaches to this course. During these past twelve years, this course has undergone a drastic transformation in content and approaches. The very name of the course has changed. The most popular label at present is Computer Literacy (to be literate about computers). Ten, twelve years ago, it was often titled An Introduction to Computers, or Computers and Society. Ten, twelve years ago, the course emphasized dominantly technical topics: Boolean algebra, number systems manipulations, and in-depth emphasis on the interior complexities of the computer. At that point in time generally only a brief reference was made to computer applications and societal implications. Such a course was required for computer science majors and minors and offered as an elective for the general student. The writer discussed these changes in 1974, in a paper titled "Computers, Society, and Man" and will not review this material in depth, but refer to this early history. (1)

Paradoxically, as the present course became oriented towards the interdisciplinary (non-computer science) student, a new course gradually emerged for the scientist. The planning for the latter course markedly resembles the early version of Introduction to Computers, with added materials on structured programming and algorithms. It has little or no reference to society.

Almost all current Computer Literacy courses briefly reference the technical arena. These courses synthesize the history of computers, introducing hardware, software, computer applications, and social implications -- and integrate these topics with a generous introduction to the use of a computer in an interactive, timesharing environment.

California State University, Chico (CSUC) introduced Computers and Society ten years ago. Many institutions were teaching similar courses. However, since that date, these early courses (here and in other institutions) have been greatly modified due to the revolutionary changes in hardware and software, resulting in user-oriented applications systems, with thousands of new applications of computers. These changes have altered professional working environments, have enlarged the realm of computer science, and they have given rise to a new person: the user. This person may or may not program -- may merely "use" a system for specific applications and purposes.

With user-oriented (or user-friendly) systems, there evolved a trend for computer education for every student. Such a revolution made it easier for a great many persons to get involved in using a computer, without becoming a computer professional. This trend has been evident for at least seven to eight years. This specific trend or thrust will continue to grow strongly and will cause a genuine metamorphosis (or Renaissance) in undergraduate education.

BRIEF COMPARISONS, IDEAS

In 1974, the writer presented a CCUC paper on the topic, "Computers, Society, and Man".(2) The opening paragraph of that paper is given here, as it is highly relevant today:

"The concern for human society and human uses of technology is one of the most vital, important, and thoughtful topics in undergraduate education. One of the main purposes of undergraduate education is to prepare the student for the future. We live in an age of growing computerization in every area of life. Yet education and the mass of human beings have not accepted the reality of the computer as a potential positive force for man and human life. Further, the effects of the computer upon every individual in a technological society are so great, that it is necessary that these human beings know what a computer is, what it is not, and how we might use cmmputers and advanced technology in the service of man. To accomplish these objectives, we need general education courses in 'Computers and Society' on both the lower and upper division undergraduate levels."(3)

Looking back at that particular paper, it is very closely related to today's Computer Literacy courses. However, in 1974, the writer's approach and methods were 4-7 years ahead of many departmental approaches to this subject. It took four years at this institution (CSUC) before these approaches were generously endorsed and became integrated with curriculum planning for Computer Literacy. It was obvious to the writer, even in 1970, that the "user" had a different frame of reference, a very different need than that of the scientist, and that Computers and Society (now Computer Literacy) was needed by interdisciplinary students from all majors.

There is one major difference between the early course and today: more programming. At the earlier date, we did not have enough timesharing systems to afford an in-depth exposure to programming, only a brief one. Again, it was evident in 1974, that concomitant programming had to be a vital part of the course.

Since that time, the writer has participated in many national and international conferences dealing with computers and their applications in education and the arts, and in working conferences of the Association for Computing Machinery (ACM) and the National Science Foundation (NSF). Because of space, only a few references precisely relevant to this paper are cited here.

Many of the ideas presented here are the result of recent curriculum planning by the author, made possible in the form of released time from teaching. Some materials are excerpts from a recent proposal to this University, submitted to the CSUC General Studies Committee in the fall of 1981. Some of the ideas presented here have been successful with recent groups of students. In working to meet the needs of interdisciplinary students, some specific techniques and approaches have been developed that go beyond the planning of other groups (MECC, ACM, NSF) for Computer Literacy.(4),(5)

COMPUTER LITERACY AND GENERAL EDUCATION

The handwriting on the wall in 1982 is very clear: in 2-5 years Computer Literacy will be required for all undergraduates. At the present time, many institutions are encouraging students to study computers and gain a broad exposure to this important mind tool in the form of Computer Literacy via General Education (or General Studies) credit. The whole realm of GE is very broad, and only a few comments are presented here.

In 1976 Harvard began an intense (publicized) review of its GE program, calling for a renewed "broad education for every student". Publication of this scrutiny by Harvard revitalized the ongoing battle of educational institutions to present a broad, relevant curriculum for their undergraduates, to produce "educated" persons.

In January 1978, Edmund C. Berkeley, editor and publisher of Computers and People, wrote an editorial, "Computers, and an Educated Man". (He did mean the educated person.)(6) In his brief, excellent, one-page editorial, Mr. Berkeley suggests that we need to revise our definition of an educated person to include: "He (she) has an elementary knowledge of computers and some experience with them." (7)

In October 1980, the Chancellor of the California State University and Colleges (CSUC System) issued a mandate for re-examination and reformation of of General Education within the CSUC System. This mandate is closely related to Harvard's original impetus to upgrade and reform GE. Other systems have since revised their GE programs. By 1981, specific departments within CSUC were asked to submit new GE proposals. Again, some of these ideas presented were part of proposals for CSUC Computer Literacy for General Education planning.

The goals of broadly educating persons to take their place in a technological society are complex and significant goals. Computer Literacy, in the author's opinion, is a cornerstone course in broadly educating persons for participation in our technological society. While colleges and universities have a charge to prepare students for varied and specific utilitarian professional careers, this is but a partial responsibility. Our colleges and universities are not mere trade schools. We are interested in the whole person -- today and tomorrow, with philosophical and societal considerations in addition to professional preparation.

Within this frame of reference, the important issues resulting from growing computer applications in varied professions, the topics of automation, changing work patterns, data banks, privacy, social responsibility, computers and legislation -- these are not esoteric areas of intellectual concern for specialists, but these are issues that will affect intimately, importantly, the lives of all citizens in our society. These are important concepts and ideas that every educated person needs to live and survive -- to participate effectively in this society.

Thus the thrust of Computer Literacy for General Education becomes more important than merely learning to use a computer. It exceeds "computer appreciation", going into areas of change, social responsibility, laws, ethics, ideas related to the computer but beyond the computer.

THE STUDENTS AND THE RATIONALE

Computer Literacy is aimed at people who may not take another course in computer science. In many cases, perhaps 60-70 percent of these students, this is their first and last computer course. The detailed listing of objectives for this course specifies particular understandings and experiences that are aimed at the interdisciplinary student who is studying computers for the first time. These students represent highly diversified majors, from literally all disciplines. Ideally, this student is at the Freshman or Sophomore level. The person who takes this course at such an early level in his or her academic career is better able to utilize computer experiences in the major in upper division work (even if not taking computer courses). At

25-

this early level, the mind of the student is more open to new ideas; i.e., the student has not yet been "programmed" by the chosen major into compartmentalized ideas of a given discipline.

As each year passes, the positions that these students will have in our world will increasingly require some acquaintance and varying levels of expertise in using the computer. In too many instances, the majority of today's undergraduates are unprepared for their future professions if they have not had at least some slight exposure to computers and their applications, their impacts.

OBJECTIVES AND OUTCOMES

As a foundation survey course, this curriculum planning has a deliberate, broad grouping of objectives. The writer would give these present objectives a shelf life of perhaps five years for undergraduate institutions -- but no more. After that period, the course will be quite changed, as discussed at the end of this paper.

The detailed objectives integrate those of the Minnesota Educational Computing Consortium (MECC) and those of ACM/NSF, from the Workshop on Computers and Society, July 1978.(8),(9) The objectives also include materials from teaching this subject for ten years, and from close contact with publishers. The format follows the MECC recommendations, since many individuals have considered it to be the most definitive description devised thus far. Because of space, it is placed after the paper, as an appendix.

CATALOG DESCRIPTION

Computer Literacy (Computer Science 10)-3 Units

Introduction to the computer for interdisciplinary students. Brief overview of calculating and computing throughout man's history. Development of problem definitions, algorithms, flow-charts, general problem-solving in the student's major. Introduction to the components of a computer, input and output devices, hardware, software, and general terminology. Introduction to Basic (or a timesharing language) with attainment of intermediate programming skills in the student's major or area of interest. Readings, discussions, and writing relate to the student's major and to general computer applications and significant societal issues. Includes one lecture, one discussion period, and one programming laboratory. Prerequisites: None.

SUGGESTED TEXTS

Dorf, Richard C. Computers and Man. (Third Edition) San Francisco, CA: Boyd and Fraser, 1982.

Boillot, Michel and Horn, L. Wayne. Basic. St. Paul, MN: West Publishing Company, 1979.

We have used many texts during the past ten years but have returned several times to the Dorf texts because they are thorough, with a fine balance of traditional computer materials, applications, and implications. We have prepared special Study Guides for this text that students have found valuable. They also use the guides for discussion sessions. In addition, multiple section courses make use of a common test bank of multiple choice tests for the text, containing materials directly relating to the text, but also a large proportion of text-independent questions for the course.

The Boillot book on Basic is also very thorough. However, we have found that slower or faster students need supplementary materials to meet their needs, and the Reference Library materials help in this area. We have a strong commitment to meeting individual needs of students, as their capacities in such a class are very diverse.

As suggested references twenty-five texts in each of three categories have been placed on Reserve Library holdings for use by Computer Literacy students. The three categories of references are:

1) Computer Literacy, Introduction to Computers and Society, very diversified approaches;

2) Basic programming, varied approaches, with a few books on Pascal and Snobol, from the very simple to the complex;

3) Background references, some classics and new works.

Because of page restrictions placed on authors for this conference, these materials are not included here. Lists of the three categories of texts and brief comments are available to readers.

In addition to these references, RAND Reports, Government Documents, etc. are also used, and students receive a special introduction in the use of these materials by Library personnel. Computers and People is the most common and popular periodical for this class, since it focuses on computer applications and societal issues.(10)

RECENT CLASS-TESTED APPROACHES, METHODS

A few brief comments on the new format for the class are presented here. We have gone to the large lecture mode, with diversified speakers on varied topics, often following the order of the chapters of the book. The classes are then broken up into small groups of 30-35 students each for two periods a week: one discussion period, and one programming lab. In this small group, students know each other, help each other, get to know the instructor and vice versa. Many institutions have used this format for some time. It is more expensive to teach, but the quality of learning is higher because of added individual attention given to each student. This is important because of the wide range of capacities in an interdisciplinary class.

Here are some class-tested ideas we have found that work. Only a brief discussion is given here.

1) Team Approaches for Programming - Experiments with students have shown that a team or group approach, substituting co-operation for competition, is a definite aid in learning programming, problem definitions, etc. The group approach, coupled with extensive team blackboard work, makes learning much easier. The students are very enthusiastic and open using this approach to programming.

2) Word Problem Banks - In too many courses, all students get the same assignments, whether they are relevant or not. These assignments are too easy for some, too hard for others. In attempting to guide students to define their own problems, many repeatedly asked for lists of suggestions. In "listening" to these students, the writer (in desperation), developed word problem banks -- long lists of word problems to program. Special lists could be accessed for a listing of the program, but no credit was given for this category of problems. Other lists were merely lists, without access to the programs for the problems. Students found this approach to be very helpful.

Students were encouraged to write their own problems, going from the simple to the complex. They could also put their own word problems on the bank, and they received extra credit for this work.

3) Varying the Programs - Once a program from the list (or problem definition) was written, the

student generated many versions departing from the first: if a program was interactive, another was written using READ/DATA statements. IF statements are explored, then FOR/NEXT loops. Dimension statements are used, subscripted variables, MAT statements, etc. Subroutines are writen for repeatable modules. Strings are added to numerical programs and vice versa, and combinations of strings and numbers are generously used. The principles of programming are emphasized, and also, the use of modular programming methods. After turning a program around within many variations (particularly in group blackboard sessions), students find they have few difficulties in writing programs in their major, and in defining their own problems. Programming in the major is strongly encouraged. In this method, the student understands what is going on in the program, thus making it easier to translate it into varying new programs.

4) Introduction to Strings - Strings are introduced almost immediately, since 60-70% of the students understand words more than numbers and need to process words. Every textbook introduces numbers first -- and much later, words. Non-mathematical students lost interest and zest with this mode. Combinations of words and numbers are more natural, and students can readily use this type of programming in their majors.

5) Programming Bank - Students think differently than professors. Even in the beginning all programs must be well documented. Many of these student-written programs focus on interactive modes that other students can use. These are maintained by the instructor and assistants, although we hope that we can persuade our Computer Center to maintain and augment this bank in the near future. Programs are coded as to level of difficulty and subject. These programs are heavily used by students along with regular LIBRARY (LIB) programs maintained by the Computer Center. The latter are interactive, discipline-oriented programs that they may use -- but cannot list to see how they are written. The student-written programs that can be listed (and run) contain more relevant examples of the types of programs students write in such a course. Textbooks do not have such programs.

6) Learning to Read Programs - The writer has always emphasized learning to read more complex programs. This approach was emphasized even more after reading David Moursund's superior ideas in Basic programming for Computer Literacy.(11) The Conway and Archer text, Programming for Poets - A Gentle Introduction Using Basic and its companion, Programming for Poets - A Gentle Introduction Using Fortran are invaluable aids for the teacher working with interdisciplinary students, particularly those who need to process words rather than numbers. These texts also emphasize heavily, learning to read programs of a more complex nature.(12),(13) For these students, a fairly high level of programming reading skills is desirable, since many of them will be working closely with programming specialists in their future professional work, rather than programming per se.

Just one brief note before leaving this section. It is very important in this class to put the students at ease regarding programming, and to discern the faster and slower people. These two groups need very individual, respectful attention, so that their needs, as well as the middle, average groups are well met.

SOME TRENDS IN COMPUTER LITERACY

In attempting to meet growing numbers of requests for Computer Literacy, publishers have been alert to the pedagogical impact and needs of Computer Literacy. A good proportion of textbooks aimed at this course feature a package of instructional materials: test banks, overhead masters, an Instructor's Manual, a Study Guide for Students, and often, mini-lectures for the instructor. Others even furnish slides. This packaged trend will continue for obvious reasons: it is a great aid for the student and teacher, and it sells books.

Almost all test banks feature true/false and multiple choice materials (with keys for the instructor). Some texts feature self-testing materials for the student. True/false questions have a 50% guess opportunity, and should be discarded. It is important to remember that students do not like machine-graded tests. They tend to cause regurgitation of materials. The teacher then has to become a master of test designing to overcome the limitations of this convenient medium. Multiple-choice questions should be text-independent and meaningful, overlooking trivial materials. This is hard to achieve. Questions need to be cross-indexed and indicate some level of difficulty.

The trend in most institutions has been to develop a common pool of questions, which are added to and modified with each semester. This requires supervision by a dedicated person(s). The trend should be to develop nation-wide testing materials, with superior materials for machine-evaluated tests and written topic materials. We desperately need to get beyond the cottage industry approach to testing.

The General Education trend for Computer Literacy is a step towards encouraging students to learn about computers. Most institutions would like to require it for every student, but the hardware to accomplish this goal is beyond their means at this time. Gradual procurement of more and more hardware is a prudent approach with today's financial crunch. However, the biggest problem (after getting the hardware -- and software) is teachers. We need very large numbers of teachers who know about computers, who will use computers wherever and whenever possible in their classrooms and their research. A few universities have undertaken ardent, creative programs to teach their teachers, but the majority of institutions are comprised of faculty who are dominantly anti-computer. A review of fairly recent National Computer Conference Proceedings attests to this thrust for teacher education.(14) We are utterly unrealistic if we focus primarily on educating students for Computer Literacy, because we must somehow find the way and the means to reach their professors.

THE FUTURE OF COMPUTER LITERACY

In undergraduate education, this mass-education of students in Computer Literacy is a temporary problem or situation. As more pre-college institutions teach Computer Literacy and programming, these students will very soon enter undergraduate institutions with more complex skills in programming and a generous exposure to computer applications, implications, etc. This trend was predicted in 1975 and is highly visible now. Thus our major problem (perhaps 5-7 years hence) will be teacher education at every level. The thrust for re-educating all teachers, not merely professors, is utterly inadequate and ineffective at the present time, caused by resistance on the part of institutions and faculty and coupled with the genuine financial problems of all schools.

It is inevitable that all teachers need to know something something about the computer. It is important to bear in mind that participants of this conference understand the significance of computers in higher education, but also to be acutely aware that the greatest proportion of faculty in higher education do not share this same perception. Perhaps our mission is to help that group of individuals to become computer literate.

Now, and in the very near future, we can perceive a domino effect: students entering undergraduate education with computer skills, requiring teachers to use computers, requiring institutions to revise and upgrade curricula to integrate the computer within higher education.

If computer education will soon be taught at lower levels of education, what then, is the future of this course? With home computers becoming more popular, with microcomputers being purchased by pre-college educational levels, with increasing development of pre-college teaching of this subject, we will not be teaching Computer Literacy. We will be teaching more societal impact courses, and more complex applications of computers as computers are absorbed and integrated within all levels of education. This includes grades 1-16. (We are assuming that graduate institutions are already well massaged by the computer.)

While we cannot predict the future, we can peer prudently beyond today, to perceive highly discernible trends and follow those trends a few years ahead. Anyone doing more than that is a genuine prophet, oracle, or psychic.

Computer Literacy of tomorrow will have another title, will be an upper division course, focusing on societal issues, legislation, changing work patterns, ethical issues -- resembling the Societal Impact of Computers courses we now have (although we don't have too many of them and inadequate textbooks for this course at the present time). In other words, Computer Literacy as we know it, will not be taught at the college/university level.

Computer Literacy in the very near future needs to be available in Learning Centers for all students. (Some rich and wise schools have this at the present time.) Publishing firms will flourish with videodisc packages for this (and other) curcular offerings. The videodisc is the new generation of computer-managed instructional systems, wedded to Personalized Instruction. Professors will publish more widely in this mode, copyright their works, get professional credit for this creativity. Materials will be device-independent, transportable at last! These systems will be cost effective due to marketing techniques and "mass" production. Cottage industry CAI may cease. At long last, some of the problems confronting computer-assisted, managed, based instruction will be moderately solved.

COMPUTERS IN UNDERGRADUATE EDUCATION

Instructors in higher education who cannot (and will not) use computers may become an endangered species. Computers will be (and are) augmented by other learning systems. Graphics and word processing systems will be as common as numerical processing. There will remain some courses where the omputer will not be used. The majority of courses n undergraduate education will have some emphasis n the computer -- but most will use the computer oncurrently with the course throughout the duration f that course. The applied arts will require expertise in using the computer. The measurable disciplines will integrate computers and computer graphics within curricula and revamp their materials and departments. New applications will continue to emerge and mushroom.

In summary, the entire fabric of undergraduate education will be altered. The format of undergraduate institutions will certainly change. The teacher will be a different type of person, a Renaissance human being, evidencing high levels of right brain/left brain integration. New forms of "personalized" instruction will bloom profusely, transcending present origins. Students and teachers will meet for different purposes than at present. Testing may be self-testing. The videodisc will be the new "book". Straight CAI will be almost obsolete.

Higher education may truly become "higher" education. Universities and colleges may assume a role that is direly needed today: lifelong learning for all the people.

If undergraduate teachers continue and extend their present progress, coupling thoroughness and inventivity with wisdom and humanity, uniting art and science in a humane vision, we could (and should) revolutionize undergraduate education. We need to bring about another Renaissance.

When that time arrives, we shall no longer have NECC. We shall meet for more exalted purposes.

REFERENCES

(1) Hertlein, Grace C., "Computers, Society, and Man," Proceedings, CCUC/5, Pullman, Washington State University, June, 1974, p. 335-343.
(2) Hertlein, op. cit., p. 335.
(3) Ibid.
(4) Klassen, Dan, "Computer Literacy Objectives from MECC," ACM SIGCUE Bulletin, October, 1979, p. 29-34.
(5) ACM Workshop on Computers and Society and Computer Impact, July 16-19, 1978, Williamsburg, VA
(6) Berkeley, Edmund C., "Computers, and an Educated Man," Computers and People, Vol. 27, No. 1, January, 1978, p. 6.
(7) Ibid.
(8) Klassen, op. cit., p. 31-34.
(9) ACM, Ibid.
(10) Berkeley, Edmund C., Editor and Publisher, Computers and People, 815 Washington St., Newtonville, Mass. 02160.
(11) Moursund, David. Basic Programming for Computer Literacy. New York, NY: McGraw-Hill, 1977, 268 p. ISBN 0-07-043565-0.
(12) Conway, Richard and Archer, James. Programming for Poets - A Gentle Introduction Using Basic. Cambridge, MA: Winthrop Publishers, Inc., 1979, 323 p. ISBN 0-87626-728-2.
(13) ________________. Programming for Poets - A Gentle Introduction Using Fortran with WATFIV. Cambridge, MA: Winthrop Publishers, Inc. 1978, 332 p. ISBN 0-87626-722-3.
(14) National Computer Conference Proceedings, AFIPS Press, 210 Summit Avenue, Montvale, NJ 07645.

==

APPENDIX

OBJECTIVES FOR COMPUTER LITERACY, 1982

COGNITIVE OBJECTIVES

1. HISTORICAL

1.1 Identify the historical sequence of counting and calculating devices that the human race used before the invention of computers.

1.2 Identify the growing complexity of calculating devices, number systems of man and their interdependency upon each preceding invention and/or idea.

1.3 Recognize that the complexity of a society and its level of technological progress depends upon its "intellectual power" and the speed and efficiency of its information processing systems.

1.4 Recognize the historical sequence of specific systems and ideas immediately preceding the computer: Jacquard's loom; Babbage's Analytical and Difference Engines; Hollerith's punched card machines; the MARK I, ENIAC, EDVAC.

1.5 Compare and identify the historical changes in specific fields since the invention and use of computers in these fields: business, medicine, law, education, etc.

2. HARDWARE

2.1 Distinguish between hardware and software.

2.2 Identify and compare specific functions of input and output devices and specific applications of these devices.

2.3 Distinguish the specific capacities of the major components of a computer: central processing unit (arithmetic/logic unit and control unit); main memory, auxiliary memory; and other forms of memory.

2.4 Identify the basic operation of a computer system; input of data or information; processing of data or information; output of data or information.

2.5 Compare and identify first, second, third, fourth generation hardware.

2.6 Identify varied methods that a person can use to access a computer: keyboard, timesharing terminal; hands-on use (mini or microcomputers); networking and the specific hardware used with these methods of access.

2.7 Determine that the basic components of a computer function as an interconnected system under the control of a program developed by a person.

3. ALGORITHMS, FLOWCHARTING AND PROBLEM DEFINITIONS

3.1 Recognize the definition of an algorithm.

3.2 Develop simple to complex algorithms to define specific problems and tasks.

3.3 Detect logic errors in an algorithm and correct these errors.

3.4 Develop flowcharts from simple and complex algorithms.

3.5 Develop definitions of specific problems that may or may not be programmed by the student, with emphasis on problem definitions.

3.6 Compare problem definitions from varied disciplines and identify similar and dissimilar definitions.

4. SOFTWARE AND DATA PROCESSING

4.1 Identify the fact that we communicate with computers through a binary code.

4.2 Identify and compare programming languages on an elementary level: machine language; assembly, Basic; Fortran; Cobol; Pascal.

4.3 Identify the need for organized data that may be used in a computer program.

4.4 Identify varied symbols that may be used as data: sounds, pictures, numbers, etc.

4.5 Identify the fact that communication is the transmission (and reception) of information via coded messages and that these messages may be in natural or synthetic languages.

4.6 Identify the fact that information processing involves the transformation of data by means of a set of predefined rules.

4.7 Recognize that a computer needs a set of precise instructions (a computer program and data) to operate.

4.8 Recognize the difference between systems software and applications software and computer libraries.

4.9 Recognize that a computer processes data by searching, sorting, deleting, updating, summarizing, etc., following the instructions of a programmer.

5. APPLICATIONS

5.1 Identify that there are many programming languages suitable for a specific application and recognize the most suitable language for several applications.

5.2 Identify the common office/administrative applications used by many firms and institutions and recognize the types of software and hardware that may be used in these applications.

5.3 Recognize specific applications of the computer in one's major and future profession.

5.4 Recognize and compare specific applications of computers in the following areas: medicine; law; education; business; transportation; government; library; creative arts and humanities; manufacturing; politics; the home.

5.5 Recognize and compare the following major types of computer applications: management information systems; information storage and retrieval; process control; general computation; data processing; computer-aided design and manufacturing; simulation, modelling, forecasting.

5.6 Recognize and identify the trend towards user-oriented applications.

5.7 Recognize that applications and implications are interdependent.

5.8 Recognize that computers can aid humans in applications that benefit from speed, accuracy, repetitiveness.

5.9 Assess computer applications of today and identify potential future applications.

6. IMPACT

6.1 Identify specific changes in the student's future profession due to computer applications and the impact on that profession.

6.2 Recognize the varied methods for restricting (and accessing) use of computer systems: identification codes, passwords, etc.

6.3 Identify and assess regulatory procedures regarding one's own file in a data bank and the restrictions on the use of universal identifiers, etc.

6.4 Recognize that computerization both increases and decreases employment and assess the trend towards change and complexity in varied professions other than one's own.

6.5 Recognize and identify major social issues: automation; semi-automation; leisure;

obsolescence; changing educational patterns; privacy and the individual; the technological imperative; social responsibility, etc.

6.6 Assess and explore specific social issues in a series of oral and written reports.

6.7 Recognize that computers are used to commit a variety of serious crimes and compare varied methods for protecting and securing information, protecting the rights and privacy of individuals.

6.8 Identify and analyze the advantages and disadvantages of a data base containing personal information on a large number of people.

6.9 Identify and compare the laws of various nations on privacy and the protection of the individual.

7. PROGRAMMING IN AN INTERACTIVE LANGUAGE

7.1 Use an interactive timesharing language to write and solve beginning to intermediate computer programs.

7.2 Understand and use varied elements of the Basic language: REM, PRINT statements; INPUT, PRINT, READ, DATA statements; GO TO, IF/THEN, FOR/NEXT statements; use of subscripted variables, dimensioning; strings; subroutines.

7.3 Analyze and understand (read) computer programs of a more complex nature.

7.4 Define and program problems in Basic relating to one's major.

7.5 Differentiate and compare objective and subjective approaches to problem-solving and programming and apply these approaches to specific problems and beginning research.

7.6 Debug and correct simple to intermediate programs in Basic.

7.7 Analyze and compare varied approaches to programming a specific problem in Basic.

AFFECTIVE OBJECTIVES

1. ATTITUDES

1.1 Does not feel fear, anxiety, or intimidation from computer experiences.

1.2 Feels confident and free about his or her ability to use a computer.

1.3 Spends extra time using a computer for self-defined tasks.

1.4 Feels confident in discussing the computer with friends and relatives.

1.5 Feels confident about his or her ability to control computers and use them creatively for the betterment of human beings and the environment.

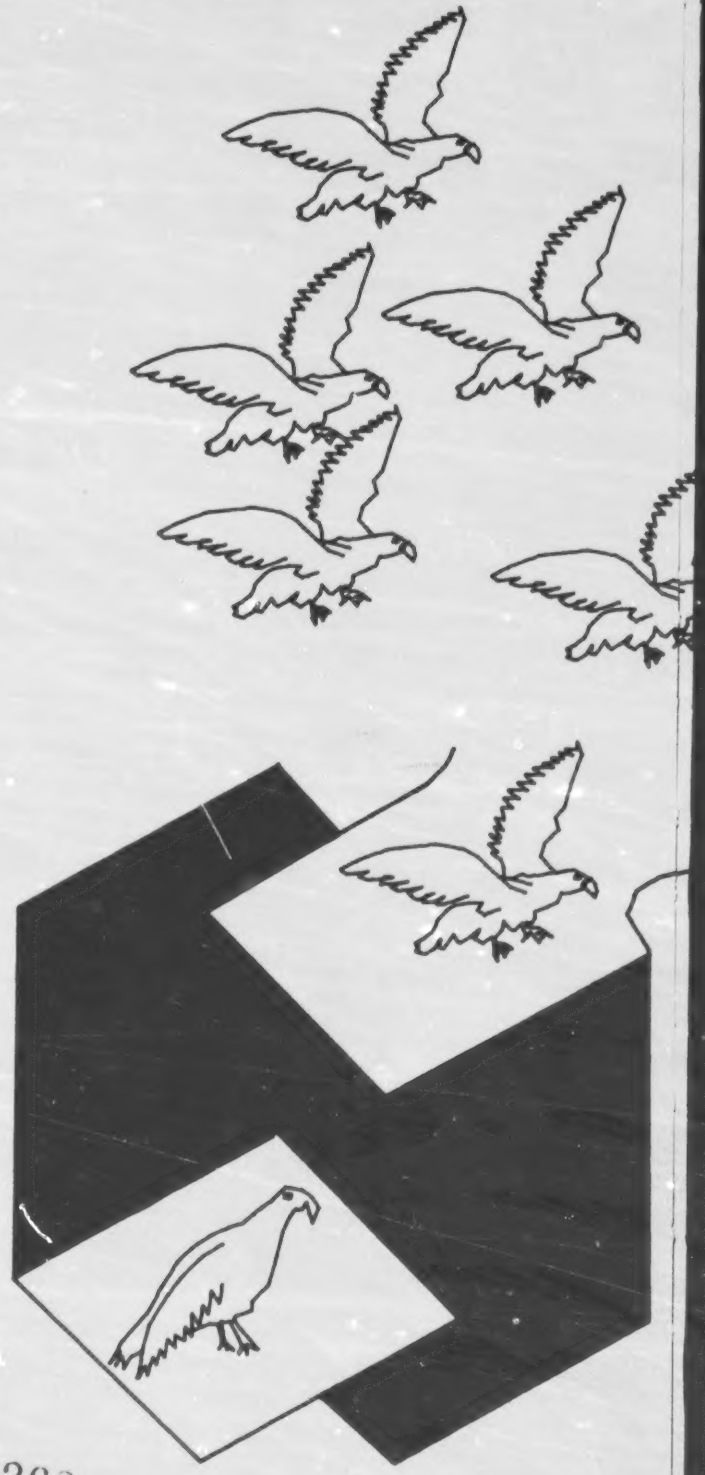
1.6 Appreciates the opportunity to use a computer system.

1.7 Values and feels confident about computers and their uses in his or her chosen future profession.

1.8 Enjoys playing with the computer on a simple or complex basis.

1.9 Feels positive about computers and the future; uses positive-affect words like challenging, stimulating, etc. in planning his or her future, professionally and personally, cognizant of the computer's increasing role in that future.

NOTE: During the past semester, we have taken small groups of students from the Computer Literacy classes and have given them a month's exposure to computer graphics. They have been taught to graph and chart, to use several user-oriented libraries for varied graphic output.



A SPECIAL COMPUTER FOR TEACHING HOW COMPUTERS WORK

Dr. J. Monk
The Open University

ABSTRACT

This paper describes a computer especially designed for education. The computer is specially simplified and has proved successful in teaching 13-14 year olds machine level programming. A part of that success is due to the motivation created by being able to directly control and operate motors and lamps within models constructed by the children.

INTRODUCTION

The Open University in the UK teaches adults in their own homes using a variety of media. For about 10 years we have provided courses in computing - almost for as long as the University has been in existence. On any one course we may have several thousand students and each must be provided with teaching materials including any experimental equipment. Because of our special requirements of high volume, low usage, the need to mail the equipment, and the need for annual testing, we frequently design or at least specify the manufacture of our own equipment. A recent course aimed at managers consists of a package of books and a small computer. The course is intended to explain how computers can be used in products. This requires an emphasis on the ability of the microcomputer to measure and control. The aim is not to teach people to become programmers nor hardware designers, but to recognize how microcomputers can be used in products and what is involved in developing those products.

Schools in the UK have also become aware of industrial applications of microcomputers in products and in production processes. A new topic referred to as 'control applications of microcomputers' is appearing in many schools. Several schools have chosen to base their courses on the Open University's small computer and my aim in this paper is to explain to those interested in the design of school equipment the features of the computer that have made it successful as a teaching aid.

THE DESIGN OBJECTIVES

The objectives of the design of the computer can be split into four parts; firstly the objectives related to teaching about what the computer is and what it can do; secondly the objectives concerned with teaching about the development process and the importance of good tools; thirdly objectives that ensure the motivation of the student and finally objectives that made the computer practical in our teaching environment.

The practicalities demanded a computer that was:

(1) small enough to mail cheaply;
(2) easy to test;
(3) cheap in volume.

The motivational aspect demanded:

(1) the computer was seen to be realistic by the student;
(2) it should not possess unnecessary complexity both in its operation and the explanation of what it is and what it does.

The teaching elements required:

(1) it should illustrate the kinds of operation that can be performed by a microcomputer;
(2) it should demonstrate a variety of data representation;
(3) it should illustrate several addressing modes;
(4) it should show how computers can be used to measure, detect, control, and display;
(5) it should demonstrate the value of development software;
(6) it should show how software can be built more quickly if ready written software modules are incorporated.

It was taken for granted that assembly language programming would be the vehicle for all practical work because the embedded applications illustrated, in an industrial context, would all be programmed in a low level language.

MEETING THE REQUIREMENTS

The seemingly conflicting requirements of simplicity and realism had been solved earlier in an Open University course entitled 'The Digital Computer'. Around 10,000 students have studied this course. Each student receives his own computer with an instruction set based on the PDP8 computer and with stack operations based on the PDP11. Its educational success for beginners to computing is due to its limited range of instructions and facilities. Little needs to be learned before successful programs can be created. The limited storage space (only 60 locations) limits the ambition of the student.

The new computer was based on the single chip processor, the Intel 8049. The 8049 is a mask-programmable device and is not normally programmable by the user. The 8049 in our computer was programmed to emulate a simplified version of itself. The software in the ROM contains the emulator, some routines to handle the peripheral devices, and a utility package that enables the user to program and control the emulation from a set of keys. The users program is held in a RAM area of the store in the 8049 chip. A student programming the emulated processor, therefore, appears to be programming a realistic commercially available processor commonly used in products.

THE ARCHITECTURE PERCEIVED BY THE STUDENT

The student has access to a computer that has a single accumulator and seven other registers. The word length is 8-bits. Instructions can be stored in 60 locations of 8-bits each and the contents of these locations can be changed by the student. Extra instructions are held in further locations but these are not alterable.

These instructions represent a set of subroutines available to help in programming the tasks set for the students. The student can inspect these locations to see what is stored there. The areas of store for data and instructions are kept separate as is the case in the 8049 itself. Input is provided either through subroutines or through a single bit port. Output is through a 6-bit port or through subroutines. The subroutine mechanism is based on a stack and the stack pointer is held in one of the registers. This feature copies the common practice in microprocessors but does prove to be a conceptual barrier that requires deliberate teaching. Simply setting the stack pointer to an appropriate value automatically when the machine is turned on would avoid the need for any deliberate action on the part of the student. Using subroutines would then become a matter of using the subroutine call. Later the role of the stack and stack-pointer could be explained.

The computer is designed so that the exploration of the various parts of the computer and its instructions can take place in simple steps with each step capable of simple demonstration by the students themselves.

The computer is mounted on a single board 200 mm x 140 mm. It is operated by a set of keys and a row of seven-segment displays. Considerable care was taken in designing the functions of individual keys and the format of the displays, although it was in this area that compromises had to be reached between extreme simplicity of operation and cost.

Programming through the keys can be done directly in binary - to show the fundamental operation of the computer. It can be programmed in denary - to demonstrate how codes can reduce the length of a written program and reduce the scope for errors (we decided it was unnecessary to introduce hexadecimal notation to explain this point). The machine is normally programmed using instruction mnemonics, like assembler mnemonics, together with literal operands which demonstrate the value of software development tools.

INSTRUCTION SET

The operations to be illustrated were a limited group that demonstrated the ability of the computer to:

(1) operate external equipment;
(2) find out the state of external equipment;
(3) perform arithmetic operations on words treated as numbers;
(4) perform logical operation on words treated as a collection of separate bits;
(5) respond to changing conditions;
(6) transfer and store data;
(7) perform subroutines.

The list of instructions chosen were one or two words in length. For output a single word OUT instruction transfer data from the accumulator to the output port. Two sets of red, amber, and green lights are operated directly by the port. Introducing this instruction first gives a direct way of demonstrating the application of the computer, the role of the accumulator, and the nature of an instruction and the program counter.

The IN instruction copies 1's or 0's into the accumulator depending on the state of a single push button. With the IN and OUT instruction, the lights can be turned on and off with the simplest of programs. The JUMP instruction enables this to be turned into a continuously operating program. Operations on data can be performed by one of three instructions: the ADD the Exclusive OR, and the Decrement instruction. Decrement is included to avoid introducing 2's complement arithmetic when the need to subtract 1 is required and also is representative of implied operand instructions. ADD and Exclusive OR have two addressing modes - immediate and direct. For example, ADD a constant to the content of the accumulator or ADD the content of register 3 to the accumulator. For conditional operation a Jump if zero (JIFO) instruction is provided. Load, with immediate and direct addressing, and store with direct addressing, allow data to be moved between the registers and the accumulator. Finally, the subroutine mechanism is provided by call and return instructions.

A total of 11 instructions with 3 addressing modes are all that are provided and all that are necessary for teaching the principles of computing at the level required by the course. The instructions are all similar in function to instructions that a real, rather than emulated, 8049 processor would perform. However, they are a small subset.

SCHOOLS

The package intended for managers in industry has been purchased by several schools. The schools have incorporated the practical work using the microcomputer into lessons. In the UK, children have a major set of examinations at age 15-16 and at age 18. Generally at age 13-14 they choose their examination subjects and continue to concentrate on the syllabus required by their chosen subjects.

Computer studies is not yet widely established as an examination subject. Teachers have a choice: to teach 13 year olds computer studies before they concentrate on their exams or to teach 17 year olds an additional subject.

My report is of schools teaching 13-14 year olds. After an hour or so of introductory teaching the children have little difficulty in getting the computer to perform simple programs, and after 8 hours are able to program their own tasks. The success and the motivation of the children is largely due to the immediacy and reality of the response from the computer. The sets of traffic lights on the board can be operated by a single instruction and a builtin delay subroutine allows simple traffic light sequences to be created. A numeric display accessible through a prewritten subroutine and the delay subroutine enable clocks and stop watches to be made. A temperature sensor and a subroutine to access its data enable digital thermometers and energy meters to be created. In one school, power switches have been connected to the traffic lights and photosensors to the input switch. The power switches are used for a variety of applications including the operation of motors from a construction set. The children can then create their own robots and machines operated by the computer. Several other schools have copied the idea of the system by simulating it on larger computers by providing a very limited instruction set, prepackaged subroutines, an elementary assembler, and the ability to control devices that move, make noises, or light up.

Providing devices that can be operated directly removes many conceptual barriers. The model in the mind of the learner of what the computer is doing does not need to hold many abstractions. Working with a keyboard and screen with alphanumerics creates problems, particularly with lower ability children,through the abstraction of the problems, the abstraction of the computer, and the abstraction of the operations being performed. The simplified computer reduces the scale of the computer model required by the pupil and the control applications remove the abstraction of the problem and to a large extent the operations being performed. All ability groups seem to achieve success in programming in assembly language and are happy to talk of stack pointers and accumulators.

CONCLUSIONS

Simplified and tailored instruction sets for educational computers reduce the burden of learning intricate detail and make judgements on the choice of instruction during programming largely irrelevant. Controlling lamps, displays, and motors - - especially when used with a construction kit -- removes the abstraction of problems tackled by the computer and can reveal the sequence of the computer's operation. With these simplifications and control interfaces, all ability ranges of children have few problems in learning about the computer and programming it in assembly language. By making the educational computer emulate a part of a real computer the skills are directly useful and the experience of learning is seen to be relevant by teachers and pupils alike.

REFERENCES

(1) Using the OPUS computer
TM221 Open University, 1973
(2) OPUS Reference Manual
TM221 Open University, 1973
(3) B du Boulay, T. O'Shea and J. Monk
The black box inside the glass box:
presenting computer concepts to novices
Int. J. Man-Machines Studies (1981) Vol.14

PROGRAMMABLE CALCULATORS IN THE MSE CURRICULUM: A REPORT ON EXPERIENCES IN THE TRENCHES

COL James W. McNulty, PhD
Associate Professor of Mathematics
United States Military Academy
West Point, NY 10996
Telephone 914-938-3200

COL John L. Palmer, PhD
Professor of Mechanical Engineering
United States Military Academy
West Point, NY 10996
Telephone 914-938-2411

BACKGROUND

Faculty opinions concerning the use of programmable calculators in mathematics, science, and engineering (MSE) courses are frequently divided between two camps. Advocates tout the potential for enhancing problem solving skills while detractors voice concern over diminishment of those same skills as a result of a "black box" approach to education. Supporters claim a savings in time which would otherwise be spent on lengthy computations and note that students can use this time to explore alternate solutions. Opponents deny such savings, pointing to student time spent in learning to program in addition to time needed to write and debug new routines. Optimists see opportunities for innovative and more effective approaches to MSE education while their skeptical counterparts see tried and true methods threatened. Both groups note the limited number of texts and lack of other material available to support use of programmable hand-held calculators (PHHC) in college level courses, and both groups seem realistic in their appraisal of the time required by faculty members to develop such supporting materials. However, detractors believe that this time will not be well spent while advocates perceive a profitable benefit-to-cost ratio.

These same pros and cons were argued when computers first began to appear on campuses. Now, given the reasonably strong acceptance of computer systems and their support of the MSE curriculum in most schools, one might expect a more positive attitude toward the use of PHHC in MSE courses. Maybe we are simply conservative by nature, or maybe our training conditions us to be cautious in any moves away from time tested techniques. Whatever the reason, we found ourselves debating these issues at the United States Military Academy (USMA) in early 1980.

For a number of years, entering freshmen at West Point had been required to purchase and use hand-held calculators. In addition, USMA has had a long term commitment to its "computer thread" - a continuing program of instruction and usage that builds and reinforces previous knowledge of computers and computer applications. However, when a proposal was made to have entering freshmen purchase one of the more powerful programmable calculators, the faculty divided fairly evenly between the two camps noted above. The previously mentioned pros and cons were argued vigorously and another dimension was added to the debate when proponents claimed that writing programs and using "canned" routines on a PHHC would support and enhance the computer thread. Opponents saw such activity as a distraction which would be counter-productive to building and reinforcing knowledge of computers.

In an effort to reconcile these differences, a study group was charged to develop recommendations on how best to deal with individual computational requirements placed on cadets by the Military Academy curriculum. Almost two dozen specific issues were defined. Among these, the following were of particular interest:

1. What should be the role of programmable hand-held calculators?
2. What should be the role of other larger machines?

266

ment instruction should be provided in the use of PHHC?

4. What is the impact of PHHC on classroom procedures, examinations, and homework requirements?

The remainder of this paper will deal with two experiments which were undertaken in an effort to respond to and follow through on these questions. It should be kept in mind that these two actions are only a small part of the overall study and follow-up activities which will be reported by other authors.

A PHHC BASED TEACHING EXPERIMENT

For several years, West Point has offered an elective course entitled Computer Methods in Engineering. It is pursued by junior and senior level MSE students who have completed courses in Fortran programming, calculus, differential equations, probability and statistics, physics, and chemistry. Cadets build on this background, working their way through seven progressively more difficult blocks of instruction covering statistical operations, root finding, matrix manipulation, solution of simultaneous equations, numerical integration, solution of differential equations, and optimization. In each of these blocks the student is required to write one or more programs, usually in the form of a subroutine. This code is validated against a set of test data and stored in a file to which both the student and the instructor have access. Once this is accomplished each student tackles a realistic problem which usually (but not always) requires the use of the computer method just studied. All work is accomplished on the USMA timeshared instructional support system which is hosted by a Univac 1100/12 computer. Several blocks of instruction require use of the pre-packaged routines available in the Univac Math-Pack library. Examinations are conducted at the terminal, to include a two hour midterm test and a four hour final. During such periods students are completely on their own to solve problems which may require application of any of the programs which they have previously written.

At the beginning of the autumn semester of Academic Year 1980-81, a group of 16 students was selected at random from this course's enrollment. Each was loaned a TI 59 calculator and printer plus master library, math, and statistics modules. These items had been purchased with funds made available by the alumni association for this special experiment. Cadets were told that, using the calculator, they would study exactly the same methods and be subject to the same examinations as their counterparts who would be using the main-frame computer. Each was given an option to change to the computer group and one individual chose to do so.

The 15 students in the experimental group began the course by learning to program their calculators. They moved from simple exercises through editing, debugging, storing and retrieving data, storing and retrieving programs, direct and indirect transfers, labels, loops, subroutines, and use of library modules. Class was conducted with students seated at small tables in a room equipped with floor outlets to which units could be connected. Text material for the course consisted of instructor generated notes with occasional assignments in and frequent reference to the TI manual, Personal Programming. As in the course which uses the timeshared computer system, homework concentrated on the development and use of subroutines to accomplish root finding, numerical integration, etc. Each of these assignments required students to present a statement of the problem, a logical sequence of steps leading to a solution, a documented listing of the resulting program accompanied by a sample run with conclusions and recommendations appropriate to the problem and its solution. These submissions were marked by the instructor with a variety of comments and suggestions and were returned to the student for appropriate corrections and retention. Each cadet was required to keep a course notebook with sections for lecture notes, vocabulary, sample programs, and corrected homework and exams. Notebooks were subject to review and grading by the instructor at random times throughout the semester. As in the computer- based course, periodic examinations required individual work in the classroom and complemented homework projects in which free and open discussion was encouraged. Faced with a set of problems, students taking these tests had to determine whether solutions required use of the PHHC and, if so, which of the previously studied methods were appropriate. They then attempted to solve the problems using subroutines which they had written.

No attempt was made to master all of the capabilities of the calculator being used. On the contrary, as with the computer-based course, the primary focus was on learning methods which could be used in the solution of a broad spectrum of MSE problems. Students were encouraged to experiment and there was frequent discussion of programming style and the art of problem solving. In previous semesters this approach had led to student critique comments about the value of a "learning-how-to-learn" experience and these remarks continued to appear in the same proportion in critiques written by students in both the calculator and computer groups.

Several developments during the experiment are particularly noteworthy, primarily because they provided the MSE Calculator Committee with information and insight which proved valuable in its study. For example, the calculator students initially lagged behind the computer group, but by the end of the second block of instruction (root finding), the calculator group had caught up and from that point on they were frequently ahead. When this happened, they simply studied an additional method. For example, in the numerical integration block, the calculator group studied Romberg integration in addition to trapezoidal and Simpson approximations. The initial lag was partially a result of time spent by the students learning how to work with the calculator while their computer counterparts were already familiar with Fortran and the local timesharing system. However, a more significant part of the lag probably resulted from the instructor's caution in moving through unfamiliar territory. Once he developed a feel for the pace at which students using PHHC could deal with the material, the course began to move more quickly. Later, when the computer-based group was slowed by reduced responsiveness of the timesharing system due to heavy demands from other courses, the freedom of "stand-alone" operation allowed PHHC students to move ahead.

The instructor had initially planned to provide some routines to cadets by recording the code on magnetic cards and distributing these cards to students. In addition, he had planned to require that cards be submitted with each homework assignment. However, while cards could be used with high reliability to store and retrieve programs for a given calculator, there were frequent problems when cards recorded on one PHHC were read by another calculator. It is assumed that this is a consequence of minor differences among machines in the alignment of read/write heads. As a result, routine exchange of magnetic cards was not pursued and this appears to have had some positive benefits. Students noted in their critiques that they could not count on using a classmate's program without keying it in themselves. For this reason they were more inclined to develop their own code from scratch. In addition, the sample code handed out by the instructor and the programs submitted by students were in the form of documented listings. This emphasis on annotated programs paralleled the emphasis on documentation in the computer-based course and appears to have caused students and the instructor to pursue a more disciplined and careful development of programs.

During the semester there were several calculator malfunctions. These were dealt with promptly by a nearby calculator exchange and repair outlet. However, the importance of having a number of calculators available for direct exchange or temporary loan became quite obvious. A student who is trying to stay abreast of material being presented in a calculator-based course cannot be without a properly operating machine for long periods of time.

The most revealing insights from this PHHC experiment began to emerge after the course was several weeks along. Student participants began to inquire about using their programmable calculators in other courses. They were encouraged to do so, provided the instructors in those courses approved. These cadets began to report on the usefulness of both the programming capability and the methods that they had studied. The most commonly expressed reason for this enthusiasm was "time saved." When queried about the use of this extra time, responses varied widely. Several students said that they "checked answers" or "tried another approach" when time allowed. Others commented on being free to move on to other problems or assignments. Six noted the special value of being able to generate simple plots quickly and believed that these plots frequently gave them helpful insight to a problem and its solution. Two quite candidly admitted that they used the additional time for extra curricular activities.

As the semester progressed, students who were not involved in the experiment but who had seen the programmable calculators being used in their classrooms or in study areas, began to inquire about the equipment and its capabilities. Several purchased the calculator and made arrangements to use printers which were on loan to cadets in the course. Instructors for courses in which cadets were using these PHHC began to seek information on capabilities, limitations, and costs. These faculty members invariably saw possibilities for some use of the programmable calculator in their course. They did not see such devices as a distraction from the pursuit of problem solving skills or as a "black box" approach to education. When asked about this they noted that they were free to limit or allow student use of the programming capability, both in their classroom and by the nature of their homework assignments. When they wanted to focus on a particular skill or technique which might have been circumvented by a "canned program", they simply guided the student or the instruction in such a way that the calculator did not interfere with achievement of the learning objectives. Several of these instructors were teaching courses which required use of the timeshared computer system. Two expressed strong belief that use of the programmable calculator enhanced student understanding of the large scale machine. Another pointed out that problems for which use of a program is appropriate occasionally lend themselves to solution with either a hand-held calculator or a larger computer. However, students seemed to have very little trouble in deciding which tool to use. When the mainframe computer was needed, students turned to it and when the PHHC met their needs, they took advantage of its availability. This should not imply that students will never have a problem choosing between computational resources, but it does seem to allay some fears that programmable calculators detract from building and reinforcing computer skills. It also seems to provide some support for having students familiar with several of the computational tools which they will encounter eventually.

Lest all the above appear overly positive, it should be noted that there is a "catch," at least in the development stage of a course using programmable calculators. The investment of faculty time in generating instructional materials is significant. The expenditure begins with the layout of the course and proceeds through writing of handouts, homework, and examinations. Teaching strategies and techniques can be different than those used otherwise. Topics covered and their sequencing may need to be changed in order to take advantage of the PHHC. Finally, the instructor must be prepared to assess and reassess the results of his labors, making changes where appropriate. This all takes time and there simply is no clear cut answer for people who ask "is it worth it?"

The instructor in the experimental course believes that his efforts were well invested in this case. His ten years of experience in teaching computer-based engineering courses brought him to this test with a number of questions concerning the proper role and the effectiveness of using PHHC in MSE courses. While some of these questions remain unanswered, he is convinced that programmable calculators can be used effectively in many of these courses and that they have a place in the array of computational resources which should be exploited in MSE education. From his perspective there were two especially important observations in this experiment. First, the PHHC students achieved the learning objectives and second, in the process, they developed increased motivation and ability to explore and learn on their own. In addition, an extensive comparison of examination results revealed no significant difference between calculator and computer-based student performance. Student critiques and feedback from other instructors were very positive. Faculty members who had previously expressed reservations were impressed with what they saw and, as some of these individuals have subsequently begun to incorporate programmable calculators in their own courses, they have found ways to deal with most of their concerns.

In spite of all this, it was clear that the test had some serious limitations and a great deal of work remained. One of the most obvious factors which influenced results was the nature of the student group itself. All were juniors and seniors who were admittedly interested in the use of computers. All had worked with calculators before and all were MSE students. Thus, it seemed important to pursue an expanded experiment with a wider student group.

WIDE SCALE INTRODUCTION OF THE PHHC- AN EXPERIMENT IN GETTING STARTED

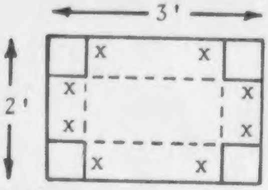
The encouraging results of the test described above and the need to gather additional information led to the decision to adopt a PHHC for use by the incoming freshman class. Although the advantages of a magnetic card memory and a companion printer were very appealing, the cost of such a combination was considered prohibitive, at least at this stage of the experiment. Instead, bids on a simple PHHC with continuous memory were sought and the TI 58C was selected.

Because students at every level of aptitude and with a wide range of interests would eventually use these machines, it seemed appropriate to introduce the PHHC early in the curriculum. Since mathematics is a subject studied by all USMA freshmen, and since there are several points in the mathematics program where PHHC instruction might enhance the learning process, responsibility for this instruction was undertaken by the Department of Mathematics. The way in which this phase of the experiment is being accomplished will be outlined in the balance of this paper. It should be kept in mind that the instruction is spread throughout the course rather than being lumped into a single block.

The goals of PHHC instruction during the introductory portion of the mathematics program concentrate on development of student competence and confidence in the use of the PHHC as a simple calculator. Once they are familiar with the keyboard and have used a number of the calculator's capabilities, an effort is made to have all students enjoy success at entering and executing a simple program. Since the first lesson of the calculus course traditionally emphasizes the flavor of calculus and its preoccupation with changing quantities, it presents an excellent opportunity for students to begin using the programming capability of the PHHC to evaluate the changes. A set of locally produced notes introduces basic calculator use and diagnostic procedures. Students work through a number of homework exercises and, at the subsequent classroom attendance, questions and misunderstandings are addressed by instructors before proceeding.

A typical problem from those used to generate discussion on the nature of calculus is shown below. By modeling the problem algebraically the class develops the following equation:

Volume = $6x - 10x^2 + 4x^3$



Find the volume of an open box made by cutting squares of side x from the corners of a 2' X 3' piece of cardboard and then folding upon the dotted lines. Find the value of x which produces the maximum volume.

Numerous observations about functions, mathematical modeling, the limitations of algebra, and the nature of calculus may be based upon repeated evaluations of this equation and a graph of the function. After using simple calculator steps to generate data for a plot, students begin to see the similarities between these evaluations. This opens the door for writing and entering a simple program which facilitates rapid evaluations of the function. The appeal of the PHHC as a labor and time saving device is firmly established by the end of this lesson.

The next phase of instruction offers repeated opportunities for students to use the PHHC and to systemize what they are learning about it. Functional evaluations are required in a variety of contexts from sketching to solution of word problems. In addition, some students find it useful to write programs which compute slopes and intercepts of lines, the length of a line joining two points, angles between lines, etc. Advanced students pursue topics which lead them to other PHHC applications such as finding the nth Fibonacci number. It may be of interest to note that achievement of mathematical learning objectives did not appear to be enhanced by using the PHHC to supplement instruction on limits. In some cases, instabilities are introduced by round off. These inaccuracies prove to be misleading when exploring the existence of certain limits and this necessitates much classroom discussion which is not really germane to calculus. However, this early encounter with an inherent limitation of digital computation is valuable to the student.

The final PHHC activity for this introductory phase is built around determination of equation roots using both the bisection method and Newton's method. An adequate bisection program would require conditional branching so emphasis here is on the use of the bisection routine available in the Master Library module which comes with each TI 58C. Newton's method, on the other hand, provides the opportunity for a major application of subroutines Throughout this root-finding block of instruction the PHHC is used to support the preliminary work of making sketches and establishing intervals on which roots exist, in addition to actually writing and executing programs to find roots.

Once students are comfortable writing and using simple programs they are ready to take advantage of some of the more powerful features of the PHHC. For example, an opportunity to introduce conditional branching arises very naturally in conjunction with definite integration. In this phase of the instruction, all students are required to program the Trapezoidal Rule in sufficiently general form that the function, the limits of integration and the number of subdivisions of the interval of integration may be varied by the user. They are not required to program Simpson's Rule as it is already available in the Master Library Module. However, comparisons are made between these two integration approximations and the efficient and intelligent use of such readily available tools is a major item of emphasis. Advanced students receive additional exposure to conditional branching during the development of fundamental concepts of definite integration. These students write and use PHHC programs to evaluate upper and lower Riemann sums for certain well behaved functions.

The final phase of PHHC instruction is built around vectors. Initially, students write programs which let them store a set number of two or three component vectors. Next, addition and scalar multiplication operations are programmed. Finally, scalar products and vector products are taken and stored. The tangible differences between vectors and scalars and between the scalar product and the vector product become very real to students when they find that they must provide storage space for one or the other. This use of the PHHC appears to enhance student understanding of vectors and vector manipulations while reinforcing programming abilities.

The instructional program just described is spread over two semesters. Work is also underway to take advantage of the power of the PHHC in follow up mathematics courses which offer a wide variety of opportunities for further applications and reinforcement: definite integrals of functions of several variables, series, numerical solution of differential equations, solution of linear systems, and numerous statistical uses. To further encourage exploitation in science and engineering courses, members of appropriate departments are being informed of the goals, accomplishments, and problems encountered during these initial experiments. Their response has been encouraging, with every indication that acceptance and application will be wide spread.

Although the experiment is not yet complete, student interviews and instructor reports indicate that most instructional goals are being achieved. There is a general high level of enthusiasm by students and instructor enthusiam has been growing as they find that PHHC use can actually facilitate the teaching of mathematics, thus justifying the time and effort required to plan and implement PHHC based instruction. Motivation of both students and instructors has been enhanced by inclusion of programming problems on examinations. In retrospect, it might have been advantageous if all students had been taking their Fortran course concurrently with the initial PHHC instruction but this has not been the case. During first semester, half of the freshman class was taking such a course and during second semester the rest of the students will study Fortran. No comprehensive survey has been made yet but anecdotal reports indicate that some students perceive a beneficial relationship and mutual reinforcement between the courses. This matter will be investigated further.

CONCLUSION

It seems clear that the bottom line has not been written. Only as this freshman class approaches graduation will it be possible to assert confidence in answers to the questions listed earlier in the background section. Our tentative answers are beginning to take shape. For example, the delineation of roles for the large computer and the PHHC appears to occur naturally. They frequently complement each other and students seem to thrive on the availability of an additional computational resource. Indeed, learning to decide and choose the better tool for a particular job should be viewed as a worthwhile result of MSE education. It seems clear that the instructor can control PHHC impact on classroom procedures, examinations, and homework. In fact, it is hard to imagine a situation in which a question cannot be phrased or a requirement cannot be specified in a manner which will cause the student to pursue the learning objective defined by the instructor. On the other hand, in order to take advantage of the PHHC as a teaching tool, faculty members must be willing to invest time and effort in phrasing those questions and specifying those requirements. The key lies in each instructor's knowledge of the course material, his understanding of the capabilities of the PHHC and other resources, and his ability to guide his students appropriately. In this context we believe that the two experiments described above indicate that the PHHC can play an important role in MSE education.

HELPING TEACHERS TEACH ELEMENTARY MATHEMATICS WITH MICROCOMPUTERS, CALCULATORS, AND MANIPULATIVES

Margaret L. Moore
Science and Mathematics Education
Oregon State University
Corvallis, Oregon 97331
(503) 754-4031

INTRODUCTION

Mathematics is more than computing, knowing, and/or committing to memory that 9 + 1 = 10. More importantly, it encompasses an understanding of why 9 + 1 = 10. The field of mathematics is more than a set of basic facts; it includes processes by which the basic facts are discovered.

The question arises as to how children learn mathematics. Current learning theories suggest that learning mathematics is an evolutionary process which must begin with the student's existing comprehension. Learning progresses as the student relates his/her own understandings to new structures. The process requires the active involvement of the student, at least mentally if not physically, as well as a motivating environment for the student. Zoltan P. Deines maintains that learning mathematics is not a "spectator sport"; that is, the student needs to interact with a variety of concrete embodiments of mathematical structures. The student must be allowed to explore,organize, and abstract mathematical structures. In its initial stages, however, learning mathematics requires concrete objects,an active, hands-on, experiential approach, a manipulative approach.

In reality, mathematics is taught in a strikingly different manner than the process suggested by learning theorists. Elementary mathematics classes are directed by an adopted text and workbook rather than the needs of the child. Children work quietly at their seats, raise their hands to ask the teacher, but do not share answers with each other.Manipulatives, lab activities, and games are reserved for Fridays or "fun activities to do after the worksheets are completed and the learning has taken place." This approach appears to signal that learning mathematics does not happen through these activities. Such an approach emphasizes mathematics as a set of facts rather than a process, a problem-solving process.

In recognition of the growing discrepancy between how children learn mathematics and how children are taught mathematics, the National Council of Teachers of Mathematics (April 1980 annual meeting) recommended that problem solving be the focus of school mathematics of the 1980s. Additionally, they recommended that mathematics programs take full advantage of the power of calculators and computers at all grade levels.

One of the fastest developing technologies impacting education is the computer/calculator technology. Innovations in both hardware and software are giving impetus to the development of a wide range of educational software. At the June 1980 annual meeting of the National Educational Computing Conference, a progress report was presented for the Association for Computing Machinery's Elementary and Secondary School Subcommittee. The committee, aware of the massive influx of computers into the nation's schools, focused on the "computer illiteracy" of teachers. The committee noted that there was (1) little use of computers and calculators as aids in learning mathematics and experiencing problem solving and (2) little integration of computers and calculator activities into the mathematics curriculum at the elementary level. Although the technology is developing at a rapid rate,opportunities for enhancing teaching of elementary mathematics with the use of computers and calculators have not been explored because most elementary teachers have not been trained to integrate them into classroom activities.

A local expression of the national needs and concerns with technology and the mathematics curriculum occurred at Oregon State University (OSU) in the fall of 1980. With new budget expenditures, a large number of Oregon schools received computer hardware, usually at least one microcomputer.

Typically, the hardware was delivered to teachers who were computer illiterate. As a result, many districts formed computer steering committees or hired specific individuals to address the problems of integrating the technology in the curriculum. Many of these groups and individuals approached the School of Education at OSU for assistance with teacher training programs directed toward teaching with computers and calculators in specific content areas. Their requests may be summarized as follows: Help our teachers incorporate the potential offered by computers in a way which capitalizes on ways children learn specific concepts.

In response to the requests, the Department of Science and Mathematics Education collaborated with the Department of Elementary Education to develop a workshop. The project was directed toward the improvement of teaching of elementary mathematics (K-6) by integrating computers and calculators with other manipulative activities. The creation of the workshop was guided by the following:

1) Learning mathematics requires concrete objects which embody mathematical structure; children need variety, an active, hands-on, experiential approach

2) The incorporation of computer and calculator activities with other manipulative activities requires a variety of activities as well as careful selection and sequencing of those activities; a mathematics classroom devoted solely to computer activities is as sterile a learning environment as a mathematics classroom devoted solely to completing worksheets

3) Teachers need a support group to help implement a change in their teaching style and to integrate calculators, computer and manipulative activities; training groups of teachers from a school/district not only provides the support group but also provides a group to train other teachers in the school/district

This proposed project, Infusing Microcomputers, Calculators, and Manipulatives Into Elementary Mathematics, was funded by a grant from the U.S. Department of Education, Pre-College Teacher Development in Science program. The grant provided an intensive three-week workshop (July 1981) for 46 elementary (K-6) mathematics teachers or administrators, plus three academic-year follow-up sessions (November 1981, February 1982, April 1982).

PROJECT OBJECTIVES

The emphases during the workshop and follow-up sessions were as follows:

1) Encourage elementary teachers to explore the teaching of mathematics in terms of activity-oriented approaches with emphasis on problem solving

2) Provide elementary teachers with a basic computer literacy

3) Expose elementary teachers to use of the microcomputer as an activity-oriented approach for learning mathematics

4) Acquaint elementary teachers with various models depicting a carefully-sequenced variety of activities (computer, calculator, and manipulative) for teaching specific mathematical content

5) Assist elementary teachers in the preparation of specific models using computer, calculator, and manipulative activities for their specific grade-level, mathematical topics

THE WORKSHOP

The three-week workshop was divided into two phases: (1) orientation and (2) infusion. The first phase--five days, eight hours a day--was held on the OSU campus to orient participants to the manipulative, activity-oriented approach to learning mathematics. Mary Laycock, a nationally known elementary mathematics specialist and elementary mathematics teacher, enthusiastically helped the teachers experience the exploration of mathematical concepts using manipulatives. Attention was directed to purposeful play as a means of entering the child's world of learning mathematics. The teachers were actively involved with manipulatives, games, and activities which stimulate and encourage children's learning of mathematical concepts of number, numeration, measurement, geometry, and sets and logic.

Teachers were taught the manipulative approach to learning mathematics by experiencing the manipulative approach. They were involved with activities with base ten blocks as concrete embodiments of the concept of place value. They experienced the importance of place value for an understanding of the arithmetic operations of addition, subtraction, multiplication, and division. They used fraction tiles and discovered why 1/3 divided by 5/3 is 1/5. They made a concrete representation of the problem with fraction tiles. Teachers were exposed to mathematics as more than rules and facts.

Throughout the week, Laycock stressed the need for the following:

1) A variety of activities, games, and manipulatives

2) A careful selection and sequencing of the activities

3) Careful attention to development and integration of computer and calculator activities among the variety of manipulative activities

The second phase of the workshop, a two-week session, was aimed at preparing the teachers for infusing computer and calculator activities with the manipulative approach to learning mathematics. Each day was divided between laboratory sessions and presentation/discussion groups. For this phase, the workshop was moved to Corvallis High School to use two classrooms of Apple microcomputers (10 per room) and adjacent classroom facilities.

At the beginning of the workshop, the teachers were totally unfamiliar with computers and thus potential instructional uses. Any work with computers needed to begin with available computer software and evaluation of software. The method was by doing evaluations. The teachers evaluated games, tutorials, simulations, drill and practice, and problem solving software. Could they use the software? Would they use the software? Should they use it? Did the software promote understanding of the ideas rather than rote memorization of facts? Would the software make a helpful addition to the variety of ways of presenting a concept at a concrete level? Could the software be modified and adapted?

Teachers interacted with a cross-section of commercial software. In addition, special software was specifically designed to be incorporated with the manipulative approach to teaching mathematics. This software was designed to demonstrate the utilization of the computer's capability to enhance transfer from concrete, manipulation of objects to mathematical abstractions. Concrete objects were graphically represented and were, thus, slightly more abstract representations of the concept than the physical object. During the previous week, the teachers had used base ten blocks for understanding place value and addition concepts. Through the set of computer programs, BLOX, representations of the physical blocks were used. In successive programs, pictorial representations of the blocks were placed next to the numeral representation for the number. Place value was emphasized by trading blocks and observing the simultaneous trading procedure with numerals. Another program dealt with estimation of sums by having the user estimate and input a sum of a set of numbers. Then the computer requested the sum from use of a calculator. The user was to then input that value. In this manner the computer was used with another manipulative. One computer program directed the activities of the user with another manipulative, fraction tiles. Using pictorial representations of fraction tiles, the program helped the user solve fraction problems with his/her own fraction tiles. The use of graphics coupled with immediate feedback and other concrete manipulatives introduced the teachers to a new method for teaching mathematical concepts.

In order for the teachers to be willing to use computers in teaching mathematics, they needed to understand computers as well as computer programming. Small laboratory groups were used to introduce the teachers to the microcomputers, to rudimentary concepts of programming in Basic, including the process of computer programming. Each group was given directed work with the concepts plus daily practice and experimentation sessions to try the ideas.

The major topics covered in these sessions during the two weeks included the following:

1) Computer operation
2) The computer programming process as a problem solving process: creating, coding, running, and debugging
3) Basic commands: PRINT, LET, IF-THEN, TAB, GOTO, FOR-NEXT, and READ-DATA
4) Apple low-resolution graphics

All of the topics were approached through problem solving. The teachers experimented with the ideas through varied problems (not exercises). Excitement, enthusiasm, and anxiety were all present. By the end of the first week of these sessions, frustration was definitely present. The teachers formed groups to attack the problems. They sensed the importance of working together, sharing, and helping each other. This activity was noted as being disallowed in their own classrooms.

Introduction of low-resolution graphics during the second week provided a significant turning point in teacher confidence. When the Basic programming concepts were focused on a more concrete problem--a picture to be programmed--the teachers began to feel in control of the machine. They drew pictures and manipulated those pictures by adding to or subtracting from them. They pictured gardens growing, seasons changing. At this point, they developed for themselves the idea of subroutines. No formal work was planned on subroutines, but the teachers needed subroutines and discovered the usefulness on their own during practice sessions. Their final programming assignment was to write a drill program which required the use of other manipulatives or calculators.

The most common comment about these two weeks of programming dealt with a clearer understanding of the problem solving process. They experienced the difference between doing problems and doing exercises. They also recognized the importance of active participation and interaction, the importance of sharing ideas in a classroom setting. They needed lots of concrete activities to understand programming. They learned from games.

Presentation/discussion seminars were distributed throughout each day to discuss methods of teaching with computers, calculators, and manipulatives. As teachers interacted with software, such as BLOX, discussions focused on incorporating the software with other manipulative activities in a careful sequence of activities. During these sessions teachers were introduced to models of carefully-sequenced activities for teaching specific mathematical content. Their assignment was to design a model which described activities and sequencing for a particular concept they planned to teach. The activities were to include games, concrete-object activities, calculator activities, and computer activities.

FOLLOW-UP SESSIONS

The final phase of the project was devoted to evaluation. It focused on implementation of workshop ideas in each teacher's class. The November 1981 session stressed classroom organization. Teachers shared problems as well as successful implementation efforts. Organizational strategies were discussed. During this session, teachers were shown another model for integrating computers, calculators,and manipulatives in teaching mathematical concepts. The goal of this model was to develop an understanding of division. Base ten blocks were used to show division as repeated subtraction; calculator activity used a calculator with only addition and subtraction keys; students were to find the quotient through repeated subtraction. Finally, students were to write a computer program to have the computer divide by repeated subtraction.

The goal of the second follow-up session was to re-energize the teachers with the workshop ideas. Teachers had reached the mid-year slump, were tired, and were less enthusiastic about continuing to make changes in the way they taught mathematics. They were beginning to return to the textbook and worksheets. Therefore, this session emphasized a review of some of the activities from the workshop. The teachers worked with base ten blocks and fraction tiles again. They played some games and did some calculator activities. Programming activities included a look at a unit to teach computer programming at the elementary level through problem solving.

PRELIMINARY EVALUATION

The guidelines for selecting teachers to attend the workshop gave preference to groups of teachers from schools/school districts. A number of the teachers attended the workshop as a group. In one case eight of the eleven teachers in a single school attended the workshop. During the final evaluation, the advantage for these groups was apparent. The teachers indicated that the support gave continued encouragement for incorporating and trying new ideas.

Although the analysis of evaluation data was not completed at the time of this writing, preliminary information indicates the following:

1) Teachers gained a more positive attitude toward computers, calculators and manipulative activities

2) Teachers significantly changed their plans to teach mathematics integrating a careful sequence of computer, calculator, and manipulative activities

3) Teachers significantly increased their knowledge about computers, calculators and manipulatives

These areas were assessed in the final follow-up session to determine the long-term effect of the workshop ideas. The information will be available in May 1982.

SUMMARY

The purpose of the project was to help teachers teach mathematics the way children should learn mathematics,that is, in activity-oriented, experiential manners using concrete embodiments of mathematical concepts. The challenge was to encourage teachers to incorporate the potential offered by computers and calculators with other manipulative activities. Overcoming computer illiteracy provided a beginning. Learning by active participation was a tremendous lesson for the teachers.

THE EFFECTIVENESS OF MATHEMATICS INSTRUCTION IN THE CAI MODE: A COMPARATIVE STUDY[1]

By

Babur Mustafa Pulat
Department of Industrial Engineering
North Carolina A & T State University
Greensboro, NC 27411
Tele: (919) 379-7780

Chiang Wang[2]
Department of Systems and Industrial Engineering
University of Arizona
Tucson, AR 85721
Tele: (602) 626-1543

ABSTRACT

This paper presents the experiences of the authors in teaching a freshman remedial mathematics course through two different methods: (a) conventional and (b) CAI (Computer Assisted Instruction) at North Carolina Agricultural and Technical State University. The two methods are compared through a nested experimental design in terms of effectiveness in preparing students for the succeeding calculus sequence. The results give evidence of the CAI mode being at least as effective as the conventional format. Furthermore, the CAI mode may generate better prepared students as opposed to the conventional format (risk ≅ 14%). This implies that all sections of the remedial mathematics course may be taught through the CAI method thus freeing some manpower to be utilized in other areas of mathematics. The details of the experiment, and the experiences at A&T's School of Engineering with Computer Assisted Instruction are discussed.

INTRODUCTION

The purpose of this study is to evaluate the effectiveness of teaching the freshmen pre-calculus course for North Carolina A&T's engineering students in the CAI (Computer Assisted Instruction) mode as opposed to the conventional method of teaching (lecture mode). It is hypothesized that the CAI format of instruction in the remedial math courses may be at least as effective as the conventional format. Furthermore, the CAI mode may produce students better prepared for the succeeding calculus sequence. If these hypotheses can be supported by experimental evidence, then all sections of the remedial math course may be scheduled under the CAI format thus freeing some manpower resources to be utilized in other areas of mathematics.

During 1980-81, the four engineering departments (architectural, electrical, industrial, and mechanical) at North Carolina A&T State University had 740 black students out of a total of 999 undergraduate students. Only 5-6% of the undergraduate students are white U.S. citizens. The foreign nationals come from 32 different countries. The U.S. citizens represent a number of school systems within or outside state of North Carolina. Thus, every year the Engineering School is faced with the problem of working with freshmen from disparate backgrounds, and even with those who lack sufficient mathematical skills to complete an engineering program. The latter is partically caused by the University's general philosophy of encouraging admission to the University in recent years. On the average, 50% of the freshmen admitted to the Engineering School are advised to take a remedial non-credit mathematics course - Math 110, which is a review of high school Algebra, Geometry, and Trigonometry. Math 110 is designed to make up the students' deficiencies in mathematical skills required for successfully completing an engineering curriculum. It is also hoped that Math 110 will help the engineering faculty later work with students with uniform high school mathematical skills rather than disparate skills.

THE COMPUTER ASSISTED INSTRUCTION AT A&T

The School of Engineering at North Carolina Agricultural and Technical (A&T) State University submitted a proposal to the General Electric Foundation in May 1978. The purpose of the proposal as

---

[1] This study was partially supported by the General Electric Mainstream Funds.

[2] Dr. Wang was a member of the faculty of Industrial Engineeing at NCA&TSU during the time of the conduct of this study.

stated was, ".....to bolster the quantitative skills of our minority engineering students with pre-engineering deficiencies." A Computer Aided Learning Laboratory was to be created in the Engineering School for pre- and early engineering academic development. In the Fall of 1978, the grant was approved along with the first year's budget.

In September 1978, actions to initiate Computer Assisted Instruction began. During the period from Fall 1978 to the end of Spring 1979, sixteen computer lines were connected to the campus DEC System 1070 (now replaced by 1090) mainframe; the needed disk drive (RP-06) was installed; the CAI software was leased and implemented on the DEC System; two 8-line groups of DC-104 communication ports, sixteen terminals, laboratory tables, and chairs were purchased and installed.

The software (CAN CAI) was leased from the Ontario Institute for Studies in Education (OISE). In short, the CAN CAI software currently in use at A&T is composed of two models. ~~P-MATH~~ (Pre-requisite Mathematics) reviews basic algebra.(1) T-Math (Technical Mathematics) consists of mainly trigonometry and geometry.(1) Appendicies 1 and 2 give the topic lists for P-MATH and T-MATH, respectively.

Typically, in a conventional review course such as Math 110 at A&T, the instructor must cover a wide range of subjects at a uniform rate. The students may or may not need intensive instruction on a number of these areas. Some may already possess the required skills on certain topics. Thus, on every topic the instructor may be boring a group of students while not covering the material in sufficient detail for those who need extensive help on that topic.

The P-Math module of the CAN CAI software tests for proficiency in each objective (there are several objectives under each topic) after providing sample instructions. If a student demonstrates sufficient knowledge in that area, the computer lets him/her quickly progress to the next section. Otherwise, it provides extensive instruction and then cross-examines the student vigorously. If the student fails to demonstrate a grasp of the concepts, he/she is logged off with instructions to seek assistance from the instructor who is monitoring the lab. Thus, students are provided with effective and individualized instruction. It is this desirable characteristic (concentration on the weak topics of each user) of the CAN CAI software that signals its probable effectiveness over the conventional method.

North Carolina A&T's experience with the CAN CAI started in the summer of 1979. The PREFACE program, designed to bolster the mathematical skills of students who would enter as freshmen in the Fall 1979, was held in CAN CAI format for five weeks in the Summer 1979 in an intensive fashion. Flower and Craft(2) report that the program resulted in increasing the basic mathematical skills of the participants. This increase was quantified through the administration of the math portion of the CTBS before and after the program. On the average, there was a one grade level increase in basic math skills.

Before discussing the experimental procedure of our study, it will be beneficial to review briefly the format of the CAN CAI for Math 110. Each student is self-paced. In order to get a passing grade, a student must complete both P-Math (average 30 hours) and T-Math (40 hours). In a typical semester (Fall or Spring) there are 6 scheduled CAI exposure hours per week. There is always an instructor or a graduate student available during scheduled hours for student reference. The laboratory is open from 8:00 a.m. to 5:00 p.m. on weekdays and for 19 hours during weekends. Student tutors are available to answer questions during the night hours and all weekend hours. Students are always urged to follow the courses at a uniform rate throughout the semester. Sometimes dealines are given for completing certain sections.

THE EXPERIMENTAL PROCEDURE

During the Fall 1979 semester, three sections of Math 110 were offered. Two were in conventional format and one was in the CAI format. The students were randomly assigned to these sections. We followed the performance of these students in the subsequent (Spring 1980) calculus course. As previously stated, the aim was to evaluate the comparative effectiveness of the CAI format and the conventional format in preparing students for the first calculus course.

In order to test for statistical significance of preparedness through conventional or CAI format, a nested experimental design is utilized. This allows testing for calculus and Math 110 effects. It is the Math 110 effects that are of concern for this study. The dependent variable is the semester average (0-100) at the end of the first calculus course.

RESULTS

As discussed in the previous section, the experiment has two factors: Math 110 (the remedial math course) nested under Calculus I. Three sections of Math 110 with different instructors were offered in Fall 1979, two of which were taught with the conventional mode and one with the CAI mode. Four sections of Calculus I were taught by different instructors in Spring 1980. The overall semester averages in Calculus I of those students who took both Math 110 and Calculus I were used as the performance measure in our analysis. The complete design structure and number of students involved in the first phase of the study are shown in Table 1.

Since the objective of this study is to evaluate the teaching effectiveness of CAI mode versus conventional mode, a preliminary analysis were conducted to determine if students from the two conventional sections of Math 110 performed significantly differently under each section of Calculus I. The idea was to combine the two conventional Math 110 sections under each Calculus I section if the difference did not come out to be significant. The test was carried out using the NESTED procedure in SAS.(3) The results showed that the performance difference in Calculus I between the two conventional Math 110 sections is not statistically significant at the 55% level. Therefore, students in the two Math 110 conventional sections were pooled together under each Calculus I section. The revised design structure then consisted of two sections of Math 110 (CAI and conventional) nested under four sections of Calculus I. The design was again analysed using the NESTED procedure in SAS. The ANOVA table for this analysis is given in Table 2. The F value of 0.714 suggests that even at 35% significance level, the CAI versus conventional mode of instruction in Math 110 did not crate significantly different levels of preparation. The negative results could be partly due to severe imbalance of sample sizes in the data set, and partly due to small samples in each cell.

TABLE 1. THE DESIGN STRUCTURE AND SAMPLE SIZES OF THE FIRST PHASE OF THE STUDY

CALCULUS I	MATH 110	NO. of STUDENTS	POOLED DESIGN STRUCTURE
SECTION 1	Section 1	11	21
	Section 2	10	
	Section 3 (CAI)	7	7
SECTION 2	Section 1	4	8
	Section 2	4	
	Section 3 (CAI)	5	6
SECTION 3	Section 1	0	3
	Section 2	3	
	Section 3 (CAI)	1	1
SECTION 4	Section 1	5	12
	Section 2	7	
	Section 3 (CAI)	3	3
	TOTAL	61	61

TABLE 2. ANALYSIS OF VARIANCE TABLE FOR THE REVISED NESTED DESIGN - PHASE I.

Source of Variation	SS	D.F.	MS	F
Calculus I Sections	11712.96	3	3904.31	11.224
CAI vs. Conventional Modes in Math 110	989.07	3	329.69	0.714
Error	17083.44	37	461.71	

In order to solve these problems, a second replication of the study was performed. Specifically, students passing the CAI and one conventional section of Math 110 course during Fall 1980, were followed in the subsequent calculus course (Spring 1981). All attributes and variables of the first phase of the study were kept the same (controlled) for the replication, except two. First, the responsible person to assign grades to the CAI group was different. This posed no problem since the students were exposed to the same instruction set (P-Math, and T-Math), and the same evaluation criteria as before. Second, the contents of the first calculus course were changed slightly for that academic year. However, we looked at the performance of only those students who took calculus under the same instructors. In fact, we could get more data only under two sections of Calculus I for this reason. Nevertheless, this approach eliminated instructor effects, keeping content change effects to a minimum.

Before the second phase of the study could be performed, we looked at the possibility of pooling together the data across the time scale (years), and then, across the two conventional sections of Math 110 again. Results showed that there were no significant differences. Thus, aggregations were performed and data analyzed again using the NESTED procedure in SAS. The design structure, and the ANOVA table for the second phase (Phase I plus the replication) of the study are given in Tables 3 and 4 respectively.

The F value of 1.82 indicates that the preparedness of students coming from conventional versus CAI Math 110 is close to being significantly different ($\alpha \cong 14\%$). The cell means suggest that in general the CAI mode prepares students better for the first calculus course. Thus, one would be taking 14% risk if he/she scheduled all Math 110 sections under the CAI mode. Although this result may not be breathtaking, with the addition of 17 more cases during the replication, a major step has been taken toward statistical significance achieving a more balanced experimental design structure. Compare the present 14% risk factor with 40% after the first replication. At present, we are waiting to see the results of the third replication of the study.

TABLE . The Design Structure and Sample Sizes of Phase 2

Calculus I	Math 110	Pooled Data Points	Mean Performance
SECTION 1	Conventional	26	49.59
	CAI	9	64.29
SECTION 2	Conventional	8	21.62
	CAI	6	22.00
SECTION 3	Conventional	9	33.33
	CAI	6	39.33
SECTION 4	Conventional	12	16.13
	CAI	3	49.86
	TOTAL	78	

TABLE 4. Analysis of Variance Table for the NESTED Design - Phase 2

Source of Variation	SS	DF	MS	F
Calculus I Section	15627.05	3	5209.02	9.93
CAI vs. Conventional Modes in Math 110	3812.02	4	953.00	1.82
Error	34099.56	65	524.61	

CONCLUSIONS

Results supported the hypothesis stated in the introduction; i.e. the CAI format of instruction in the remedial Math course is at least as effective as the conventional format. Furthermore, the CAI mode may be producing better prepared students for the succeeding calculus sequence. This conclusion is supported by experimental evidence of preparedness through the CAI mode of instruction as opposed to the conventional mode.

One caution: We observed a higher failure rate (about 30%) in the CAI mode than in the conventional mode (about 10%) in 1979. The major reason is that the CAI program is self-paced. A typical student starts the program with enthusiasm, but looses it under pressing demands from other courses. He feels like since he has time until the end of the semester to complete the program, he can concentrate on other courses and leave Math aside. However, as the semester progresses, the available time which can be utilized for the CAI Math 110 course decreases due to the increased demands from other courses and computer breakdowns. Final efforts fall short of meeting the criteria for completing the program. Only those students who have already developed the skill of allocating sufficient time to each course throughout the semester complete the program. Having observed this fact, during the subsequent offering of the CAI-Math 110, we have introduced motivational factors into the program, such as giving frequent exams and setting target dates to reach certain objectives. These efforts have proved to be beneficial to pull the average student out of the relaxed mode.

Another factor which needs to be addressed is the cost of operating the CAI Laboratory. In addition to the required student assistants to run the laboratory and to provide assistance to the students, a considerable amount of computer time and overhead (e.g., maintenance) is also needed. However, the establishment of a CAI Laboratory opens the door for other disciplines and areas to explore this new teaching method. Also, the result of this study has been shown that the additional operating cost can be justified due to students' improved performance.

REFERENCES

1. Evannah, J. Sakamoto: The CAN/CAI Course Users Manual, Department of Measurement, Evaluation, and Computer Applications, The Ontario Institute for Studies in Education, Ontario, Canada, 1979.
2. Flower, K. W., and Craft, W. J.: The Use of Computer Assisted Instruction Techniques as a Freshman Screening Tool. Engineering Education Journal, February, 1981.
3. Barr, A. J., Goodnight, J. H., Sail, J. P., and Helwig, J. T.; A User's Guides to SAS. SAS Institute Inc., Raleigh, NC, 1976.

APPENDIX 1. P-MATH TOPICS LIST

1. Fundamental Operations with Signed Integers
2. Facoring of Integers
3. Handling of Signs and Common Factors
4. Equivalent Common Fractions and Reduction to Lowest Terms
5. Complex Fractions
6. Fundamentals of Decimal Notation

Review Test #1

7. Operations with Decimal Numbers
8. Percentage
9. Laws of Exponents

Review Test #2

10. Algebraic Substitution
11. Addition of Monomials
12. Multiplication with Monomials and Polynomials
13. Division with Monomials and Polynomials
14. Monomial Common Factors
15. Multiplication of Polynomials

Review Test #3

16. Special Products
17. Simple Linear Equations in One Unknown
18. Ratio, Proportion, and Variation
19. Systems of Linear Equations

Review Test #4

20. Quadratic Equations
21. Scientific Notation
22. Significant Digits

Review Test #5

23. Logarithms*

* There are two versions of this topic available. The student would take either four-place or five-place logs, not both.

Review Test #5 (Continued)
24. Graphing
25. Advanced Graphing
26. Formula Manipulation

Review Test #6

APPENDIX 2. T-MATH TOPICS LIST

1. Estimation

Calculator Test #1
2. Exponents and Radicals
3. Operations with Radicals

Calculator Test #2
4. Radian Measure and Conversion
5. Applications of Radian Measure
6. Trigonometry I
7. Trigonometric Functions
8. Trigonometry II
9. Complex Numbers
10. Albegraic Fractions
11. Variation
12. Algebraic Fractions
13. Trigonometric Equations
14. Properties of Logarithms
15. Exponential and Logarithmic Equations
16. Determinants
17. Matrices

BIOGRAPHICAL NOTE

Babur Mustafa Pulat is an Assistant Professor of Industrial Engineering at North Carolina Agricultural and Technical State University. He joined the University on a permanent basis in 1979. Dr. Pulat received his B.Sc. and M.Sc. degrees from the Middle East Technical University in Ankara, Turkey. He completed his Ph.D. progran in the Industrial Engineering Department in North Carolina State University at Raleigh, NC in 1980. During his doctoral studies he was partially supported by NATO.

Chiang Wang is presently an Assistant Professor of Systems and Industrial Engineering at the University of Arizona. He received his Ph.D. from the University of Iowa. His major research interests are statistical analysis, computer graphics, and mathematical programming.

Special Session

ALTERNATIVE TEACHING TECHNIQUES: A POSSIBLE SOLUTION TO THE CRISIS IN COMPUTER SCIENCE EDUCATION

Sponsored by: SIGCSE
Chaired by: Nell Dale
Univ. of Texas at Austin

ABSTRACT

The crisis in Computer Science Education is clearly spelled out in the following statistics from the snowbird report, November 21, 1980.

The total number of PhD Computer Science faculty in the US increased from 805 in 1975 to 825 in 1979. The net gain, 20, is 1.7% of the total of 1130 PhD's graduated in the same period.

The total number of PhD graduates, the next generation of researchers and teachers, has decreased from 256 in 1975 to 200 in 1980.

In 1980, fewer than 100 of these PhD's chose academic careers, and they had 600 academic positions from which to choose.

Undergraduate enrollments have doubled since 1975, while faculty size and lab space have remained nearly fixed.

Class sizes are significantly higher than in other science departments.

At the Computer Science Conference in St. Louis in February, 1981, there were 34 job offers for PhD and 12 for each Bachelors candidate.

These figures come as no surprise to those of us who teach undergraduate computer science courses. If these courses, which absorb so much of our faculty and TA resources, can be handled in other ways, then these resources can be reallocated to handle the increase in advance courses.

Participants

Hal Berghel
University of Nebraska at Lincoln

Henry Etlinger
Rochester Institute of Technology

Stephen Franklin
University of California at Irvine

Billie Ann Rice
DeKalb Community College

Adele Seeff
University of Maryland University State

Special Session

MICROCOMPUTER EDUCATION FOR TEACHERS: THREATS, TECHNIQUES, AND TRIUMPHS

Sponsored by: ACM SIGCUE
Chaired by: Herb Nickels
Calf. State College at San Bernadino

ABSTRACT

With the proliferation of microcomputers in precollege institutions, a new challenge for our higher education system is posed: the preservice training and inservice training of teachers in incorporating the computer into the classroom. While most colleges and universitites offer BASIC programming courses, many of these courses do not meet the needs of the classroom teacher.

The panelists will explore some of these unmet needs such as relevant program examples and assignments, instruction in non-computer vernacular, training in the basics of BASIC, hands-on-experience, instructors experienced in precollege classrooms, lists of hardware and software resources, and education in the design and development of instructional computing materials. The successes and failures of training programs on large, small, and remote university campuses will be discussed.

Participants

Ron Adams, British Columbia
College of New Caledonia

George Culp
University of Texas at Austin

Project Presentation

UNDERGRADUATE NURSING CURRICULUM
USE OF COMPUTERS

Chaired by: Reese Clark
Temple University

Sue Grobe
Ruth M. Neil
John R. Tharp
Barbara Thomas

ABSTRACT

An Authoring System for Developing Computer Assisted Instruction Programs on Nursing Process

Susan J. Grobe, RN, PhD, Asst. Prof. and Project Dir. (CAI), University of Texas at Austin, School of Nursing, 1700 Reb River, Austin, TX 78701

The paper reports on the first year of a three year project "Developing Authoring Models for Computer Assisted Instruction on Nursing Process," funded by the Division of Nursing of the Department of Health and Human Services (D10 NU26044-01).

The purpose of the project is to develop and evaluate an authoring system for nursing faculty's use in constructing software based on the five steps of nursing process (i.e., assessment, diagnosis, planning, intervention and evaluation).

The instructional programs written by faculty using the system will be simulations of each of the nursing process steps. The programs on each of the steps can be used as individual modules, or as a combined sequence of all five steps.

Use of the authoring system by faculty for creating the programs does not require knowledge of either computer programming or instructional design. Nursing faculty will be prompted for nursing content from any clinical practice setting, in order to design the simulations for use by students. Use of the developed programs will help students in learning to apply nursing process. The resulting instructional programs can serve as supplements or prerequisites for clinical nursing practice.

The prototypes for the authoring modules on nursing diagnosis and nursing assessment will be developed by late June. The prototype models will be presented as a part of this presentation.

ABSTRACT

Course Description for EDUC 872, Readings in Education

Ruth M. Neil, Ft. Hays State University, Hays, KS 67601

Objectives:

1. Adapt a small unit of study from the current FHSU Nursing curriculum for use as a computer-assisted instruction module.
   a. Design an appropriate computer program to provide student interaction with the module content.
   b. Explore the possibility of utilizing a simulation format for portions of the module.
   c. Incorporate a "branching" pattern in some aspects of the module organization.
   d. Employ the use of visual aids (photographs, graphics, charts, etc.) by invorporation of 2 x 2 colored slides and the use of the random access slide projector interfaced with the computer.
2. Collaborate with other nursing faculty.
   a. Select the content to be developed so that it will correspond with the School of Nursing long range plan for transition to computer-assisted instruction.
   b. Validate presentation-of-content plan with other faculty for accuracy of information.
   c. Share experiences in designing a computer-assisted instruction module as part of continuing faculty development.
3. Produce the module using the Honeywell Level 6 minicomputer (including the random access slide capability) located in Stroup Hall, FHSU School of Nursing.
   a. Collaborate with the audio-visual

department in production of appropriate 2 x 2 colored slides.

b. Consult with the Honeywell Corporation as necessary re: entering the program utilizing Can-8 language.

Submitted January 26, 1982

ABSTRACT

The Role of the CAI Author as Curriculum Developer--What Instructional Design Theory Can Tell Us About Planning CAI

John Tharp, Research Assoc., CAI Project,
School of Nursing, University of Texas
at Austin, Austin, TX 78712

This report will highlight the computer based instructional module authoring system being developed at the U.T. School of Nursing. This system is being developed from a basis of instructional design theory and systems design theory.

The emphasis of this report is on new guidelines from instructional development modeling that can be used to guide the planning and writing of instructional computing software. Until recently, very little attention has been focused on creating educational software that is designed from an instructional standpoint first, then implemented on a computer.

Participants will be briefed on Bloom's model of instruction, which calls for specification of Student Entry Characteristics, Delivery of Quality Instruction, and Assessment of Learning Outcomes. Examples from the project will be used to illustrate how each of these broad areas can be broken down into discrete planning units.

Handouts will be available that summarize the authoring process from an instructional perspective, and outline the use of this process to make development of programs easier, faster, and more instructionally sound.

ABSTRACT

An Orientation to Computer Literacy in Nursing

Barbara Thomas, PhD, Professor or Helen
Nan Hoozen, MA, Instructional Designer,
College of Nursing, University of Iowa,
Iowa City, IA 52242

The goal of giving appropriate attention to computer applications in nursing has taken the form of computer literacy in the curricula. In the undergraduate program, it is expected that all of the nursing core courses and many of the electives in nursing will include computer applications such as the following: 1) computer-assisted instruction is expected to be implemented in each course, 2) entry/retrieval of patient records are skills needed by nurses, 3) nurses should also have familiarity with administrative uses for nursing service and nursing education and 4) data analysis. In the past, some baccalaureate nursing students have been involved in computerized literature searches and a few have worked with faculty research projects.

In the graduate program, candidates for masters degrees are required to complete theses and most involve computer analysis of data. In addition, an enlarged research course sequence is being implemented this year to provide opportunities for graduate students to learn how to 1) prepare data for computer analysis, 2) create, maintain and modify data sets, 3) analyze data utilizing statistical packages and library programs, 4) use consultant services appropriately, and 5) use computers to produce tables and graphics.

A slide-tape program is being developed to serve as an introduction to computer literacy. This project presentation will outline the objectives and procedures used in developing this program. An excerpt of the slide-tape presentation will be included.

ABSTRACT

Improving Instruction in Nursing Research

Barbara Thomas, PhD, Prof., College of
Nursing, University of Iowa, Iowa City,
Iowa 52242

A proposal to develop CAI applications for nursing research was recently awarded support from WCC's Call for Proposals. The purpose of this project is to develop at least one CAI lesson for each of the five core courses in the baccalaurate program, and two CAI lessons for the graduate research methods courses. The titles for the five core courses include the following:

I - Introduction to Research Terminology
II - Critiquing Nursing Research Reports
III - Conceptual vs. Theoretical Frameworks
IV - Data Collection Instruments
V - Preparing and Analyzing Data

These programs are primarily tutorial. For example, in the terminology lesson, definitions of ten common terms are given along with examples from current nursing studies. Evaluation is built into the program by means of application questions for each term. In the Critiquing lesson, students are given a reading assignment and a worksheet for recording relevant notes. When they sign on, they are presented a systematic approach to critiquing along with questions which require evaluation of the assigned article. At the conclusion of the program, the number right is reported; if this score is 80% or better, the student is given a choice of five articles for testing without guidance. If 80% mastery is demonstrated, the critiquing assignment is complete. If not, supplemental reading assignments are given and the student must choose another article for another attempt at reaching the mastery score.

These programs are being developed on the PRIME system. Ray Far Huang, a doctoral student in Instructional Design is the programmer.

Project Presentation

ELEMENTARY SCHOOL USE OF COMPUTERS

Chaired by: Alan Roecks
Education Service Center

Sandra Y. Belton
George Burns
Jerry Chaffin
Michael Cook
Barbara Dubitsky
Marcia Krause
Sister Barbara Kushan
Marilyn Ruggles

ABSTRACT

Reading Skills Courseware Series (c 1982, Scott, Foresman and Company, Electronic Publishing Division)

Sandra Y. Belton, Reading Courseware Developer, Electronic Publishing Company, 1900 E. Lake Ave., Glenview, IL 60025

The Reading Skills Courseware Series includes twelve learning modules designed for use on the Texas Instruments 99/4A or 99/4 microcomputer. Accompanying each module is a teacher's guide that provides reproducible worksheets developed to reinforce skill learnings presented on the module, and a reader that offers a variety of selections related in skill areas, and, often, in content to the module activities.

The twelve modules in the series combine to provide instruction in the major areas of the reading curriculum: word identification, comprehension, study and research, and literary understanding and appreciation. There are two modules directed to each of the traditional grade levels one through six, each module focusing on three specific skills.

Each module includes three tutorial and three practice sections and one application assessment section. The structure and design of each module facilitate its ease of incorporation into all reading programs, including those for ESL and mainstreamed students.

ABSTRACT

Case Studies of Children Learning LOGO

George Burns, Classroom Teacher, Bank Street School for Children, Instructor, Bank Street College of Education, 610 W. 112th Street, New York, NY 10025

We are fascinated by the variety of paths children have taken in learning and using LOGO and by the depth of their involvement with this work. Our observations of children have taught us a great deal about what they are capable of doing with computers and what their interests are. We have also learned new things about the children themselves.

We will present several case studies of eight and nine year old children who have been using LOGO for a year or more. With varied achievement levels, skills and interests, each child has engaged in computer activity which is challenging, as well as personally significant.

ABSTRACT

The Role of The Classroom Teacher

Michael Cook, Classroom Teacher, Bank Street School for Children, Instructor, Bank Street College of Education, 610 W. 112th Street, New York, NY 10025

At Bank Street children are learning to program computers using the LOGO language. The children actively discover the properties of the computers and the language through the pursuit of their own project ideas. This approach has put considerable demands on teacher expertise and time. Among other things, it requires the teacher to be constantly aware of what each student is doing on the computer in order to know when it is appropriate to teach new skills. It means the teacher must know enough about programming to connect a child's own project with some catalogue or hierarchy of skills and to pull out the appropriate one for that child and that project. It necessitates an impromptu lesson which may take the teacher away from other children for some two to fifteen minutes, depending on the difficulty of the skill being taught. It expects of the teacher a great deal of creativity and flexibility in directing children toward projects and in advising them on strategies

to carry out those projects. And it demands some very high level debugging skills on the teacher's part, and a fair amount of time, to analyze children's programs which aren't working to their creator's satisfaction.

ABSTRACT

Teaching LOGO in Child-Centered Classrooms

Barbara Dubitsky, Education Coordinator, Center for Children and Technology, Instructor, Bank Street College of Education, 610 W. 112th Street, New York, NY 10025

As part of a research project at The Bank Street College of Education, children in two classrooms in the Bank Street School for Children are learning LOGO. Since April, 1981, there have been six microcomputers in each of these classrooms. The children use the computers either along or in pairs throughout the day as scheduled by their teachers.

Bank Street School for Children has a child-centered approach to education; curriculum is developed around the needs of the group. Many aspects of the work are individualized to meet the needs of the members of the class. The computer work is treated in the same manner. Children work on their own projects; teachers help them to develop their ideas and solve their individual problems. Both teachers and peers are the source of new ideas. Teachers sometimes introduce new material to all or some of the children. Children are encouraged to work together and often learn new material from each other.

ABSTRACT

Matching Computer-Based Reading Instruction to Preschool Gifted Children's Modality Strengths: A Whole-Brained Approach to Learning

Marcia L. Krause, 1145 Third Street,
Baraboo, Wisconsin 53913

In a paper presented at the 1981 ADCIS Conference, E. Paul Torrance, a noted gifted educator, discussed the implications of whole-brained theories of learning and thinking for computer-based instruction. As he reiterated, the main point of hemispheric brain functioning and learning is that hemispheric brain functioning emphsizes differences in processing; i.e. different styles of learning and thinking. Such a model provides a foundation for a new appreciation and assessment of individual differences. Thus, the rationale for this project was that the acceptance of different ways of knowing (left and right hemisphere strategies) as methods of learning provides the first step toward building a balanced school program.

The eighty preschool children who are in the project were identified as gifted and/or creative by means of the following instruments: PPVT, Good-enough-Harris Drawing Test, PRIDE (Preschool Interest Descriptor--a parent questionnaire), and nursery school teacher evaluations. Reading aptitude was assessed by means of the Test of Early Reading Ability (TERA). A computer-based learning-style assessment instrument was developed and pilot-tested using an aptitude-treatment-interaction paradigm. By means of this measuring tool, the author was able to determine whether a child was a left-hemispheric or right-hemispheric visual, auditory, and auditory/visual learner of reading skills; i.e. differentiating graphic symbols, decoding letters to sounds, and sequencing letter-sound combinations (words). Based on this information for each child, appropriate computer-based instruction programs have been designed to challenge the preferred hemisphere while encouraging him/her to exercise the other hemisphere.

The computer hardware involved the following: Apple II Plus microcomputer, two disk drives, Panasonic 19-inc colored monitor, Panasonic dual-channel tape recorder, Silentype printer, and Mountain Hardware clock calendar card.

ABSTRACT

A Humane and Forgiving System for Beginning Programmers in BASIC

Sister Barbara Kushan, Mathematics Dept.,
Saint Mary College, Leavenworth, KS 66048

An interactive syntax-prompting and syntax-checking system for a subset of ANSI BASIC has been developed on the APPLE II microcomputer for use by beginning programmers of age twelve and up. The subset of ANSI BASIC includes the BASIC statements DATA, END, FOR, GO TO, IF, INPUT, LET, NEXT, ON GOTO, PRINT, READ, and REM and uses numeric and alphanumeric constants, variables and expressions. As the program is typed, each statement is syntactically checked character-by-character as it is entered and the programmer receives immediate response by acceptance of the character or by an audible sound indicating rejection of that character at that point in the statement. The correct syntax of the statement is displayed in Backus-Naur form below the programmer's entry for reference and reinforcement. Additional help is available throughout the system with production language descriptions of the Backus-Naur form nonterminals and messages indicating syntactic or semantic errors in common areas for beginning programmers. The system allows the programmer to construct a syntax-error-free program and execute it immediately from the system.

ABSTRACT

Demonstration, Field Test Results, and Discussion of Computer Software Based on

Video Game Formats

Marilyn Ruggles, USD #497, Lawrence, KS
and Dr. Jerry Chaffin, University of
Kansas

An innovative total approach to teaching basic arithmetic facts was demonstrated using some new DLM microcomputer programs.

The software involves fast action, colorful graphics and the compelling arcade game format to both motivate and teach students arithmetic facts using numbers 0 through 9.

The field testing was done in USD #497, Lawrence, Kansas. Students selected were in grades 4, 5, and 6 who had not yet achieved mastery of basic facts as demonstrated by timed test results in the classroom. Three of the four students were identified as learning disabled. The pupils played five games for each skill three times per week. By the end of the second quarter grading period, after using the games for approximately two months (though only 4-6 hours on the computer), three of the students had achieved mastery in one or both areas being worked on. Tentative conclusions are that the materials are very effective from an instruction and motivational point of view. Additional data is currently being collected and will be shared along with comments and reactions of both students and teachers.

Special Session

LANGUAGE AND MUSIC INSTRUCTION IN THE COMPUTER ENVIRONMENT

Sponsored by: ACH
Chaired by: Joseph Raben
Queens College of CUNY

ABSTRACT

Panelists in this session will discuss a variety of instructional techniques utilizing computers in languages and music. Included will be such topics as teaching modern foreign languages with a computer, teaching natural language processing to undergraduate computer science students, and computer based instruction in the arts and humanities both in the United States and in Europe.

Participants

Michael Arenson
University of Delaware

Dirk L. C. Geens, Belgium
Katholieke Universiteit

Mary Dee Harris Fosberg
Loyola University

Edward Purcell
University of Southern California

Pre-College Instruction

COMPUTERIZED INDIVIDUAL INSTRUCTIONAL PROGRAM

James M. Caccamo, Ph.D.
Robert W. Watkins, Ed.D.
Independence Public Schools
1231 South Windsor
Independence, Missouri 64055
(816) 833-3433

Special education has undergone many significant changes over the past six years since the passage of Public Law 94-142, The Education of All Handicapped Children's Act.

In attaining compliance in accord with P.L. 94-142, states and local school districts have faced many problems. These problems have been, and are being, worked out and solved. One specific area of difficulty with which all agencies are struggling is the time and cost required to develop the individualized educational programs (I.E.P.). Price and Goodman (Exceptional Children, 1980) have presented the issue well. In their article, it is reported that the average amount of teacher time in the development of an I.E.P. was 390 minutes or 6.5 hours. The major portion of that time is in the actual mechanical process of writing the I.E.P.

It is quite apparent that the Office of Special Education is also concerned about I.E.P.'s, as may be evidenced by their Bulletin #64 of May 1980. The purpose of the Bulletin was to clarify the requirements for public agencies to be in compliance with the I.E.P. provision of P.L. 94-142.

The I.E.P. is a most significant part of an appropriate educational program and those in the field of special education generally support the concept of the I.E.P. The dilemma is how to have the I.E.P. and, at the same time, utilize our teachers' expertise in actual teaching rather than consume innumerable hours in writing.

The School District of Independence (MO) serves approximately seventeen hundred youngsters with various handicapping conditions. The problem of efficient and cost effective teacher utilization in special education is critical. Our teachers must teach.

The ability to meet the dilemma, we feel, exists in the form of the computer and computer-assisted instructional objective programming.

Assumptions

In order to utilize a computer to assist in writing I.E.P.'s, a primary assumption must be made. The assumption is that there are a finite number of instructional objectives used to teach handicapped youngsters. If this assumption is true, then we can gather the objectives, store them on a computer, and call them up when needed.

If one has difficulty making the above assumption, perhaps it would be easier to make the assumption that a teacher uses a finite number on instructional objectives. Even with this assumption, the instructional ovjectives can be collected and stored.

Development of Program

In our particular situation, once the assumption was accepted, the task at hand was to compile the objectives. This was done with the help of many, including the Monroe County Special Education Cooperative in Bloomington, Indiana which became our primary resource. Other data on instructional objectives were gathered and following a review of the objectives, it was necessary to collate and number. Each objective was written to be grammatically correct with either "can" or "will be able to" as the prefix statements. This allows the objectives to be used as either present levels of performance or instructional objectives. The information in Table I gives a sample of the table of contents of the instructional objectives manual. The information in Table II shows a sample page from the manual which will provide an example of the objectives.

TABLE I
Sample of the table of contents from the Manual of Instructional Objectives

TABLE OF CONTENTS

Subject	Page
General Information	i
Consulting Services	1
Behavior	
School and classroom	2
Social	5
Personal	7
Early Childhood	
Infant Stimulation	10
Socialization	12
Language	16
Self-Help	20
Cognitive	25
Motor	30
Handwriting	37
Health	
Hygiene	40
First Aid and Medicine	41
Nutrition	42
Human Body	42
Disease	43
Sex Education	44
Hearing Impaired	
Signing/finger spelling	46
Auditory Training	47
Oral and Written Language	48
Home Economics	
Clothing	49
Family	50
Household	50
Foods	51
Language Arts	
Oral Language	53
Written Language	55
Grammar	58
Information Retrieval and Processing	61
Spelling	62
Mathematics	
Computation	
Numeration	64
Addition	67
Subtraction	69
Multiplication	71
Division	72
Fractions	73
Decimals	74
Comprehension	77
Time	80
Measurement	82

TABLE II
Sample page from the Manual of Instructional Objectives

Mathematics
  Computation
    Addition
    MAAB066 Add 1-digit numbers when presented horizontally without regrouping
    MAAB069 Add 2-digit numbers when presented horizontally with regrouping
    MAAB072 Add 2-digit numbers when presented horizontally without regrouping
    MAAB075 Add 3-digit numbers when presented horizontally with regrouping
    MAAB078 Add 3-digit numbers when presented horizontally without regrouping
    MAAB081 Add 4-digit numbers when presented horizontally with regrouping
    MAAB084 Add 4-digit numbers when presented horizontally without regrouping
  Computation
  Subtraction
    MAAC003 Compute facts through 10 with the aid of fingers or a counting device
    MAAC006 Compute facts through 20 with the aid of fingers or a counting device
    MAAC009 Recite subtraction facts through 10 from memory
    MAAC012 Recite Subtraction facts through 20 from memory
    MAAC015 Subtract 1-digit problems with regrouping, involving zero

Hardware

The I.E.P. program runs on an IBM System/34 computer. The main storage for our machine is 96K with a simgle spindle of 63.9 megabytes. The printer has a print speed of 40 characters per second.

The computer is used by the District for other purposes and the business office has several large programs such as budget and payroll. Also, there are terminals in the two senior high schools which are used to teach computer science.

The program, at this point, uses only 1/10 of the computer's storage. The data file is approximately 5 megabytes and the program fills approximately 1 megabyte.

I.E.P. Process

It is important to remember that the computer does NOT develop the I.E.P. The I.E.P. is developed by the teacher(s) and the parent(s) after a multidisciplinary team evaluation. The only intervention the computer will have is to print the end product.

Once the evaluation, the staffing, and the interpretation of test information to the parents is completed, the staff works co-operatively with the parents in developing the appropriate I.E.P. to meet the handicapped child's educational need. Instead of writing out each objective, the teacher simply writes a code number for each objective agreed upon by the parent and the team. The objectives are selected from a 167 page Manual of Instructional Objectives.

If a parent of teacher would like to use an instructional objective not contained within the manual, they write the objective on the I.E.P. With this option available, the program is intended to further assist the process and not hinder the teacher or parent in choosing a needed objective.

Once the I.E.P. has been developed, the teacher forwards the list of code numbers along with the demographics and diagnostic information on the youngster to the Central Office. The data then are entered and the I.E.P. is run in four copies. The teacher, the parent, the child's cumulative folder, and the Director of Special Education receive a copy of the printed I.E.P. Appendix I contains a sample I.E.P.

Additional Computer Services

While the I.E.P. process was the initial and primary reason for the development of the computer program, other programs were also developed at the same time to assist the Director of Special Education in the overall management of special education services.

Demographic data on each youngster in special education are stored and can be recalled by entering the assigned student number. Several necessary reports for the State Department of Education, such as the Handicapped Census and Child Count, are also programmed and made available at the touch of a button.

A list of youngsters in need of third year evaluation and class lists by school and handicapping condition are also programmed. In addition, the computerized system allows the District to select parent names and addresses by school, handicap program, and type of program. In the future, this will enable us to provide specific personnel development for parents with specific needs. For example, a workshop for parents of children in self-contained learning disabilities classes to meet the specific needs of parents in how to cope with theparticular handicapping condition at that level of severity. The program can provide information on the personnel development needs of the special education teachers and assist in meeting in-service education needs.

The system also was developed to study the percentage of I.E.P. objectives which were set and which of those were met. The system is based on the assumption that if a child is evaluated appropriately then reasonable, appropriate instructional objectives will be set for the youngster. If set, then information can be learned from the rate at which youngsters meet the set objectives. The "set-met" data can be pulled quarterly, at the semester, or annually.

Option one permits the Director of Special Education to look at "set-met" numbers by teacher, by student, and by school. This allows a close inspection of the progress of each child in his/her educational program. It should demonstrate learning rates, as well as obstacles to learning, by the child which in turn will assist in personnel development specific to the needs of individual youngsters.

Another option allows inspection of special education teachers' "set-met" percentage by building. Hopefully this will allow for evaluating total service program within a building. The teachers may be encouraged to support and/or participate in in-service with colleagues within their home school.

The program can also demonstrate instructional objectives "set-met" by school building. This allows for inspection of the percentage, or rate, of progress made by individual buildings in providing special education services. It may also assist in looking at in-service needs.

One of the more important options of "set-met" allows for inspection of "set-met" by subject by teacher. Looking at the rate of "set-met" by subject we can determine which subjects the teacher is more successful at teaching. From this it is possible to determine specific instructional in-service needs by teacher.

The system allows for inspection of instructional objectives "set-met" by subject and by building. We can inspect which subjects have a higher frequency of appearing as objectives. For example, we can see the incidence of math objectives set by building. If a building has a very high incidence of a specific subject perhaps we need to look to regular education and how the subject is taught in the building.

Lastly, the information if table 6 shows "set-met" by subjects, demonstrating the frequency of subject occurrence on I.E.P.'s

The information shown in Appendix II is a copy of the program menu which demonstrates what items are available.

Educational By-Products

There are many direct benefits to having a computerized I.E.P. system, however, the by-products are equally beneficial:

(1) Due to printing, each I.E.P. is neat and clear. The parent can easily read each objective.

(2) The I.E.P.'s have district-wide consistency and all the instructional objectives are teachable and measureable.

(3) The system's impact on evaluation. In order to arrive at a present level of performance, a more in depth evaluation which is competency based must be administered.

Summary

A system of computerization of I.E.P.'s has been developed by the Independence Missouri Public Schools. The system reduces teacher time in writing the I.E.P. and in addition, the system assists in the administrative management of a special education program serving 1,700 youngsters.

INDIVIDUALIZED EDUCATION PROGRAM

Independence Public Schools

Identification Information

NAME: I.M.A. Student
PARENT: Mom & Dad Student
ADDRESS: 1234 Somewhere St.
Anywhere
MO 66666

BIRTHDATE: 10/25/70
PHONE #: 2222222
HOME SCHOOL: 001
BEGIN. DATE: 9/10/80
9/01/80
ENDING DATE: 5/25/81
5/25/81

School of Enrollment on Begin. Date: 001
Special Education Program and Level: 05

Special Education Teacher(s): James Caccamo
Robert Watkins

Date of Conference Committee Meeting: 9/01/80

IEP to be Implemented by: James Caccamo

IEP Conference Committee: James Caccamo
Robert Watkins

Educational Services to be provided:

1. Specialized programs and services:

PROGRAM	TYPE	MINS/WEEK
LD	RES	150
BD	RES	90

2. Regular classes and programs:
All academic areas except thos listed in #1 above.

Evaluation and Review:

1. Appropriate criterion tests will be administered for short term objectives throughout the school year. Records will be kept of the criterion tests and the date of initiation and completion of each objective.

2. Diagnostic tests in appropriate subject areas will be administered as needed to determine present performance level as well as serve as a basis for evaluation.

INDIVIDUALIZED EDUCATION PROGRAM

Independence Public Schools

% OF SUCCESS

SUBJECT: Behavior
School and classroom: Individual and Group

Present Performance Level - Student can:

(BEA 018) locate assigned classroom(s)
(BEA 021) stay in classroom or other assigned place with classroom teacher
(BEA 027) listen attentively and respectfully in class

Long Range Goal - Student will improve skills in the area of: Behavior

Objectives - Student will be able to:

(BEA 003) follow classroom and school rules
(BEA 030) sit where directed
(BEA 039) enter and leave the school building in a safe and appropriate manner
(BEA 042) enter rooms in an appropriate manner
(BEA 051) when dismissed, go directly to the bus
(BEA 072) lock and unlock a combination lock
(BEA 075) use assigned locker for belongings
(BEA 096) raise hand and wait to be called upon before speaking
(BEA 102) raise hand to ask for help or answer question

SUBJECT: Mathematics
Computation
Multiplication

Present Performance Level - Student can:

(MAAD003) compute Multiplication facts through 5's with the aid of a counting device or table
(MAAD009) recite from memory facts through 5's
(MAAD015) multiply by a 1-digit number with regrouping, involving zero
(MAAD018) multiply by a 1-digit number with regrouping, not involving zero
(MAAD027) multiply by a 2-digit number with regrouping, involving zero
(MAAD030) multiply by a 2-digit number with regrouping, not involving zero

Long Range Goal - Student will improve skills in the area of: Mathematics

Objectives - Student will be able to:

(MAAD006) compute multiplication facts through 9's with the aid of a counting device of table
(MAAD012) recite from memory facts through 9's
(MAAD039) multiply by a 3-digit number with regrouping, involving zero
(MAAD042) multiply by a 3-digit number with regrouping, not involving zero

INDIVIDUALIZED EDUCATION PROGRAM

Independence Public Schools

% OF SUCCESS

Objectives - Student will be able to:
(MAAD045) multiply be a 3-digit number without regrouping involving zero
(MAAD048) multiply by a 3-digit number without regrouping, not involving zero
(MAAD063) multiply by a 3-digit number when presented horizontally with regrouping
(MAAD066) multiply by a 3-digit number when presented horizontally without regrouping

APPENDIX B

Menu: IEP
1. Build demographics file
2. Update demographics file
3. Display demographics record(s)
4. Print Student number roster
5. Build objective file
6. Update objective file
7. Print objective file
8. Display objective record(s)
9. Enter new IEP's
10. Update existing IEP's
11. Print IEP report (4 copies)
12. Print IEP report (1 copy)
13. Print IEP report (all students)
14. Display IEP record(s)
15. Print three-year re-val. list
16. Summary:School/Handicap
17. Summary:Handicap/Severity
18. Summary:School/Hndicp/Severity
19. List:Parent's Address/H-C/Type
20. List:Parent's Address/Handicap
21. Set&Met:Students/Bldg/Teacher
22. Set&Met:Teachers/Bldg
23. Set&Met:Buildings
24. Go to menu IEP2

Menu: IEP2
1. Set&Met:Subjects/Bldg/Teahers
2. Set&Met:Subjects/Bldg
3. Set&Met:Subjects
4. Compute yearly Sp.Ed./Student
5. Print state handicapped census
6. Copy objective file
7. Copy IEP file-diskette
8. Load objective file
9. Load IEP file
10. Delete records-demograph. file
11. Delete records-objective file
12. Delete records-IEP file
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24. Go to menu IEP

Bibliography

Individual Education Programs (I.E.P.'s) Office of Special Education Policy Paper, U.S. Education Department, Washington, D.C.: May, 1980.

Price, M. and Goodman, L. - "Individualized Education Programs: A Cast Study", Exceptional Children, Vol. 46, #6, 1980.

Hayes, J. and Higgins, S., - "Issues Regarding the I.E.P.: Teahcers on the Front Line", Exceptional Children, Vol. 44, #4, 1978.

Lehrer, B. and Deiher, J., - "Computer Based Information Management for Professionals Serving Handicapped Learners", Exceptional Children, Vol. 44, #8, 1978.

Killen, J. and Myklbust, H., - "Evaluation in Special Education: A Computer-Based Approach", Journal of Learning Disabilities Vol. 13, #8, 1980.

MICROCOMPUTERS AND MAXI-COMPUTERS
CAN WE COEXIST IN EDUCATIONAL COMPUTING?

LeRoy Finkel
Instructional Computing Coordinator
San Mateo County Office of Education
333 Main Street
Redwood City, CA 94063
415-363-5459

At a recent educational data processing conference attended primarily by administrative application/large computer people, six of us were asked to make presentations on the uses of microcomputers in the classroom. Our sessions and the six large computer sessions were all well attended and well received.

I left the conference somewhat shell-shocked at the "feelings" that were evident if one listened to the "table talk" and the nature of the questions that were asked of the presenters. I always knew that all was not rosy between small computer people and large computer people but prior to this conference had not realized how bad the situation really had become. Keeping in mind that I am a small computer person..., what I heard at the conference was:

1. Misinformation regarding microcomputers.
2. A lack of current technological knowledge about microcomputers by large computer people.
3. Feelings of hostility towards small computers and anyone associated with them.

You had to listen carefully to hear all this. People were unwilling to openly discuss the real issues in public. The purpose of this paper is to air the issues as I see them, discuss them, and hopefully establish some dialog between educational computer people who use large and small computers. Keep in mind, my bias will show that I am a small computer person, but my background includes large computers and minicomputers.

THE COMPARISON WITH BUSINESS COMPUTING

The November 1981 issue of Personal Computing Magazine included a column called Business Computing (1). The article discussed how some large businesses are responding to the increased numbers of microcomputers that are suddenly appearing in business offices without the blessing of the data processing department. The parallel with the education business is clear. Here are some of the salient points made in the article:

> "Dick Brown, president of Computer Store, a nationwide chain, says he is seeing ever-increasing numbers of middle managers flocking to his stores in search of a machine that can do very carefully specified tasks and is available to them at all times. Brown adds that one of the major complaints from his business customers is the intractability of their corporate dp departments."

> "Most data processing chiefs prefer to publicly downplay the corporate rift over personal computers. They say their intentions are to simply set standards, both technical and physical, so that new computer users don't end up with equipment that isn't right for the job. They add that a major concern on the corporate level is to make sure that any new machines brought into the firm are compatible with future networking plans the company may have."

> "Many outside of the dp departments, though, dismiss these statements, calling them just words

> to hide the true issue: the guarding of corporate turf."
>
> Because of these differences, "many corporate personal-computer users have grown more and more protective of their machines to make sure the desktop systems are not "corrupted" by the tinkering of company dp professionals."
>
> "They (the dp professionals) work differently. They sit down and define objectives and write flowcharts, and set up 18 man-month projections of this or that. We don't want that approach on the Apple. A solution should take a couple of days to come up with. If it doesn't work within that time, we scrap it and try something new."
>
> "Sure, there's a plus in not reinventing the wheel. But this way there is always the chance that different people will come up with different wheels..."
>
> "With the kind of speed you get out of computers these days, if you burn up a little labor redoing something a few times, you're not losing anything."

That lays out some of the general feelings for all to see; now on to the specifics.

THE MICROCOMPUTER FOR THE CLASSROOM

There was a time when large computer and minicomputer timesharing systems were the norm in classroom computing. A combination of increasing telephone charges and the new microcomputer technology has changed that norm and new classroom computer installations are installing microcomputers (2). Existing timeshare systems are still being used, but as maintenance on older computers becomes an increasing cost factor, many schools are switching to microcomputers.

The growth of microcomputer popularity is based on cost considerations, reliability factors, and most importantly, the availability of new and exciting classroom courseware (software). Many of the major textbook publishers are distributing an ever increasing variety of courseware for the popular microcomputers. They did not publish software for large computers or minicomputers. Their entry to this marketplace now adds a degree of credibility to classroom computing. Since software is such an important factor in classroom computing, the entry of the major textbook publishers is sure to bolster the growth of microcomputers in schools. Who decides on which microcomputer to buy for the classroom seems to be at issue. The classroom teachers claim it is their decision to make. Educational data processing departments claim that the decision is a "computer decision" and that the decision should be made by computer professionals. Who is right?

The purchase of a classroom computer today is not just a 'hardware' decision. Without the proper courseware, successful classroom computing will not take place. For most schools, the decision as to which classroom computer to buy should be based on the availability of appropriate courseware to meet the needs of the educational program. In general, we no longer have the time or the money to develop our own courseware. We can buy it "ready-made" today, and probably from the same publisher who publishes our textbooks. The 'hardware' part of the decision was made for us by the publisher who developed the courseware we want on only one or two computers. The brand, disk or cassette,and memory size decisions were also made by the couseware publisher.

If we accept the courseware-first thesis, then who makes the decision? This decision is more akin to selecting a textbook than it is to buying hardware. Teachers decide on textbooks and teachers should decide on computer courseware, in essence making the hardware decision as well. It is the teachers who have to be sure the courseware fits with what they are teaching in the classroom. You can't buy computers today, and then find the courseware after the fact. The chances are too great that the courseware you want is only available on some other computer.

This is not to say that classroom teachers should avoid their educational data processing professionals. Their expertise can and will be invaluable. But teachers should make the final decision on courseware and hardware.

On more than one occasion I have heard the "network" argument described earlier. Why is it necessary to connect classroom computer(s) to a large computer over phone lines? One argument says that in the future you will be able to download licensed programs from the large computer host. At this writing, there is no evidence that courseware publishers even understand this concept. Besides, the costs may be prohibitive. It will require a 45 minute phone call at 300 baud transmission rates to fill a typical microcomputer diskette. The modem requirements ($200 at each end) plus the phone charge over low speed lines, plus the errors, plus the time it takes, plus the license charges, make the entire procedure questionable. Early pioneers with the same thought have already cancelled their effort.

And what about documentation? Good courseware requires pages of documentation for the teacher and even for the student. How will that be distributed? For the most part, the price of disk-based software is fairly reasonable. For a few dollars you can get a disk with documentation and rights to make backup copies, that you can use any time you want without having to rely on the availability of a large computer. You know what your total charges are up front and the program is yours to use anytime you want it. You have the control (also mentioned earlier) which is necessary to be an effective classroom teacher.

Another network argument suggests that the administration will want to download files of information from the large computer. While that may be true, that is an administration application that can be made on the computer purchased for that purpose. We are talking now about classroom computers purchased for their use in the classroom. If the school administration feels the need for microcomputers, one should be purchased to meet those needs.

And, to bring us all up-to-date, of the four major classroom computers (Apple, Atari, Commodore, and Radio Shack), all have software available that will make them act like a dumb terminal to any host computer. If you are presently sending data to dumb terminals, you can transmit, as well, to a microcomputer. If you want the microcomputer to act as a smart terminal, to upload and download files for example, you can find software today that will also do that. The terminal software for the micros is here and now. Is the large computer software available?

THE MICROCOMPUTER FOR THE ADMINISTRATION

There is much written about the uses of microcomputers for school administration applications. Few of the articles offer enough detail for the reader to really understand what tasks can and cannot be performed on a microcomputer. Here is a sample:

> "Any File Maintenance. Anything which requires large scale record-keeping and frequent access is a strong candidate for computerization. The limitation of the microcomputer, though, is speed, when compared to its big brothers. Attendance for up to 1000 students, payroll for up to 500 employees, budget monitoring for up to 200 codes can easily be handled on a microcomputer. Beyond these numbers, the current machines are somewhat limited. " (3)

A professional data processor familiar with student recordkeeping and school budgeting would have serious doubts that all these undefined tasks could be done on a micro. And well he should!

Haugo reports, "Claims have been made at both ends of the continuum. Some say that microcomputers are capable of handling all of the management computing needs of small districts and schools, while others say they are appropriate only for instructional computing needs." (4) He goes on to say that the advantages of microcomputers are their low cost, their ease of implementation, their flexibility and the low cost of purchasing commercially developed software. On the disadvantage side, Haugo lists:

1. Few administrative programs available for micros.
2. The difficulty of meeting external reporting requirements with standalone micros.
3. Some management applications require large computers (student scheduling, large data storage) and are not amenable to small computers,
4. Reliability of data storage on diskettes is still problematic.

It is obvious that there is disagreement within the educational computer community, but no one has yet to define the terms we are using. Here is one effort:

THE HARDWARE

A. Standalone microcomputer - Typical units include 48K, 8 bit bytes of memory, CRT screens of less than 80 columns, programmed in BASIC, assembler, machine code, with optional printers and floppy disk drive(s). Diskette storage ranges from 100K bytes to 500K bytes per diskette. (1 byte = 1 alphanumeric character) The latest double density, double-sided diskette technology is talking in terms of 2M bytes per dual-disk drive. There is no point in considering cassette use in adminstrative applications. Costs in the $3000-5000 range for classroom-type micros with $4000-10000 costs for larger systems.

B. Hard Disk microcomputer - Winchester-type hard disk drives are now available for the standalone micro. Configurations start at 5M bytes and range to 20M bytes. Add $4000-10000 to the above costs.

C. Cluster/networked microcomputers - Multiple standalone micros (even of different brands) connected by wire to central hard disk(s), sharing common files, common programs, with security, backup, etc. Much like a minicomputer timeshare system.

Any of these three systems are

capable of communicating to a large computer providing the host computer has appropriate software.

THE APPLICATIONS

Which computer is best to use to run administrative applications? Maxi-mini, micro, some combination of the above? There is obviously no simple solution as school size, district and state reporting requirements, degree of application "sophistication", and what computers are already available will influence the decsion. A brief discussion, however, may clarify some of the debate.

Dr. David Moursand, an instructional computer leader from the University of Oregon, is frequently asked, "Can computers replace teachers?". His usual terse response is, "If the teacher CAN be replaced by a computer, the teacher SHOULD be!" I believe the same can be said with the maxi-micro debate. 'If the application can be done as well on a micro, then it should be.' Note the qualifying words, "as well", and therein lies the key to much of the argument. Certainly, many applications can be done on a microcomputer, but when they are only done to prove a point, or are a repetitive activity to those already done on a large computer, why bother? The same applies to the large computers. When the micro clearly does the job as well, then the reduced cost and poor use of large computer time should be a persuasive argument to remove the application from the maxi computer.

Here are seven typical educational data processing computer applications with comments on which computer might be most appropriate to do the job.

1. Word Processing - For the district or school office and for teachers to use, word processing implies electronic, screen typing, with storage, retrieval, editing, and printing capabilities. Word processing has traditionally been performed on small computers. New programs have made word processing available on mini and maxi computers. The ability to use common files generated by a data base program or a word processing program is readily available on micros and makes this application seem like a natural to be performed on the microcomputers. Circumstances may dictate otherwise, but in general, word processing (with data base added) should best be performed on a micro (configuration A, B, C above).

2. School Attendance - From weekly, to daily, to period-by-period recordkeeping, the range of needs varies with size and level of school. Some sites simply want an alphabetic list of students absent to distribute to teachers for information. Others need a system integrated with the registration process with extensive daily teacher reports, weekly summaries, and periodic state reports. The scope of the job will determine the size of the machine. One problem is the ability to develop transportable software that is usable in more than one location or state. School attendance is usually performed on a large computer, depending on application complexity. Since the task varies so much from site to site, there is no clear place for this application. The ultimate solution will combine large computer and small computer attributes. We have seen an attendance program running on a micro that receives registration information by downloading files from a mainframe (soon the registration will be done on the micro). Period-by-period attendance is then processed using a Scantron card reader connected to a floppy disk based system. Scantron attendance sheets are printed on campus by the computer. Registration changes are made on site. Five categories of absences are available, all recorded on site. The system generates periodic reports for deans, principals, parents, teachers, and district offices. The surprise was that 900 student records can be saved on one floppy disk! (configuration A above) Other attendance systems use hard-disk technology to emulate that cone on a large computer. Attendance, then, lies in the grey area between large and small computer, with the definition of terms being the determining factor.

3. Test Scoring - At the school site, teachers may simply need tests corrected right or wrong. At the district level, there is a need for district level and state level tests to be corrected, analyzed, and compared against norms with extensive reporting to other agencies.

Test scoring is frequently done by an outside service-bureau that specializes in this type of application. More and more large school districts do their own scoring, in house, on their large computers. Can this be done on a micro? Yes. Scantron and other card readers have made local test scoring possible with little investment in hardware. Two major publishers have announced micro-based classroom management systems that use card read test results as an integral part of the classroom management system. But if district research requirements are such that all data must be processed centrally, that decision may preclude using micros for test scoring. This is another grey area application with combination large and small computers probably being the best solution.

4. Student Scheduling - Placing students in appropriate classes by

choice or fiat also implies a large range of need. School size, student options, etc. will determine the nature of the task. A conflict matrix should be part of any scheduling package.

Once past the elementary school, this certainly "looks" like a large computer application. But there are micro-based scheduling programs coming on the market that appear to emulate the job done on the large computers. We have seen a 600-student junior high completely scheduled using a $300 program written for a floppy-based microcomputer. This system does not 'integrate' with the rest of the data processing needs of the district but who knows what the future will bring. For the moment, student scheduling by computer is probably best done on a large computer system.

5. Budget - There are two stages to the budget application. The first is the development or preparation of the budget. The second, the day-to-day operation of the budget. Again school size, reporting requirements, budget detail, will determine the magnitude of the tasks.

Some of the scratch work of budget preparation is possible to do using Viscalc or Supercalc-type programs so prevelent on microcomputers but not yet available on the large computers. For a district of any size, the day-to-day budget operation will not be possible on a micro as the reporting requirements, audit trails, large code numbers (14 digits), and number of accounts will overburden the typical microcomputer system designed for business. Local schools may use a micro to keep track of their own records in parallel with the district office but not to replace the accounting function performed by the maxi computer.

6. Master File - A data base application for student records. At the local school site, the need may include name and address information for the school office and complete historical records for counseling. The district may require a comprehensive master file of student information to facilitate the preparation of external reports.

For most districts this application is best done on a maxi computer. Hopefully, in the near future, local schools will be able to download those portions of the files they need for local information to a micro that will give them ready access to this information at all times. In the long run, micros should be used to gather this data and transmit to maxi computer that will do the processing.

7. Payroll/personnel records - The personnel file requirements will parallel those of the student master file. The payroll requirements will vary with size of school or district. In both cases, external reporting requirements will be important.

While the gathering of hourly information, etc. may be performed on a micro, this application is obviously a "downtown" operation best performed on a large computer. Some small businesses are able to use micros for payroll applications.

CONCLUSIONS

The premise of this article is that there is disagreement within the education community over the use of microcomputers both in the classroom and in administrative applications. This disagreement has created some very hostile feelings that have interfered with sensible decision making. I have tried to show that there appear to be clear areas of use that should be controlled by teachers using microcomputers and other areas that are clearly maxi-computer applications. Many applications still lie in the grey area where maxi and micro should work together to perform the task best. That is why we must end this "conflict" or "hostility": so we can work together.

For most schools, microcomputers are taking over the classroom computing scene and doing it well. That decision is clearly a software-based, instructional decision, and should be left to the instructional experts who, hopefully, will ask the local technical experts for technical advice. But let us not stall that decision by raising unnecessary concerns. Administrative applications are not clearly defined and thus harder to place on micro or maxi. For large districts, budget, payroll, large data base applications, and large applications in general, fall clearly in the domain of the maxi computer. If a maxi is already in place in the school or district, these applications clearly will stay on that machine. If a decision to change computers or to buy for the first time is in the making, then the situation is not so clear. As micros become more powerful with larger disk storage capabilities and multi-terminal, multi-tasking capabilities (configuration B and C above), they will play in increasingly larger roll in admistrative data processing for small to medium size schools and districts.

Perhaps the most important decision to make is for the parties of both sides, micro and maxi, to lower their swords and agree to work together to provide the best data services possible to the client schools. For most administrative applications, this will likely be done using a combination of micros and maxis that transfer files back and forth over communications lines.

MICRO-MAXI REFERENCES

1. Personal Computing, Nov. 1981. Business Computing column. Pgs. 23-28
2. Electronic Learning, Nov/Dec 1981. "The Mini and the Micro- Which one is Right for You?". Susan Prince. Pgs. 44-
3. The School Administrator, April 1981. "What You Need to Know about Microcomputers." William Brosnan. Pgs. 32-33.
4. AEDS Journal, Summer 1981. "Management Applications of the Microcomputer. Promises and Pitfalls. John E. Haugo Pgs. 182-187
5. Microcomputing, December 1981. "Devasting Truth About Minis". Wayne Green Pgs. 6-8.

A REPORT ON COMPUTING IN IOWA HIGH SCHOOLS

Ted Sjoerdsma
Computer Science Department
The University of Iowa
Iowa City, IA 52242
(319) 353-5266

INTRODUCTION

During September, October, and November of 1981, I visited 75 of 80 Iowa high schools within a 60-mile radius of Iowa City, IA. The purpose of these visits was to ascertain the level of computing in the high schools, to recognize the difficulties that exist in the overall computing program, and to project the future impacts of high school computing at the college and university level.

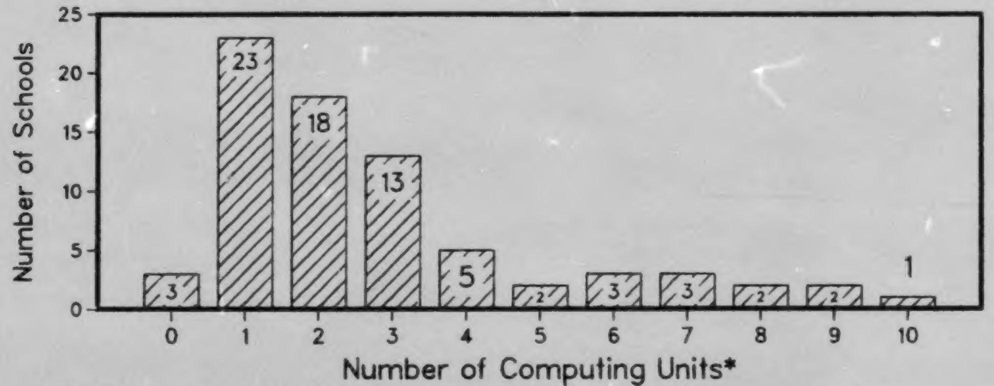
Most of the school districts visited were rural but urban schools were visited in Bettendorf (population, 24,300), Cedar Rapids (109,000), Davenport (99,800), Iowa City (47,700), and Muscatine (23,200). The school districts ranged in size from 260 students in kindergarten through twelfth grade to 20,155. The median school district size was 853. Some urban districts included three large high schools. Other districts centered in a small town struggling to maintain one small high school.

I learned much from these visits. Last fall, like many of my colleagues at colleges and universities, I thought that about half the high schools had some instructional computing capabilities and that about 25% were teaching a one-semester introductory course. I expected that over the next decade a gradual increase of computing instruction would occur in the high schools and that our best attack upon any problems that would become evident through this study would be through the teacher training courses in the colleges of education. Most of these preconceived notions had to be discarded! Several new insights were gained and these I will share on the following pages.

DATA

Equipment -- Microcomputers, such as the Apple II and TRS-80, dominate the high school scene with a few instances of terminals connected to a computer (e.g., HP3000) at a remote site. Only three of the 75 schools I visited (and possibly the five schools I did not visit) had no

Computing Units* Reported in 75 High Schools



* Computing unit is a microcomputer or a remote terminal.

instructional computing equipment. Thus 90% of the schools in the sixty mile radius have computing equipment.[†] At the 75 schools there were 203 microcomputers or terminals (125 Apples, 24 TRS-80s, 17 others, 37 terminals). Eighteen schools had 4-10 units, 57 had 3 or less units, and 3 had none.

Formal programming instruction was conducted at 55 of the 75 schools: 36 of these had a semester-long course, 19 had a full year of programming, and 16 used individualized projects for a second or third semester activity. A typical course was taught in the Basic language and covered topics such as: history of computing, hardware, variables, flowcharting, loops, functions, arrays, and subroutines. Normally 8 to 10 program solutions to problems were required of the students in a one semester course. The average class consisted of 14 juniors or seniors.

Teacher preparation in 48 of the 75 schools had no formal foundation, i.e., had no formal training such as a course which was part of a degree program. Teachers in 26 schools had completed formal computer language courses (usually Basic, sometimes Fortran or Cobol). Five teachers had completed minors in computer science. Most teachers had been involved in summer workshops or inservice training in computing.

The computer was being used as a tool in other disciplines. Students were solving mathematical problems on the computer. Chemistry students were running simulations. Social studies students were using some statistical analysis routines. Agriculture classes were using best outcome simulators on farm activities. The following graph gives indications of volume of use in the respective disciplines.

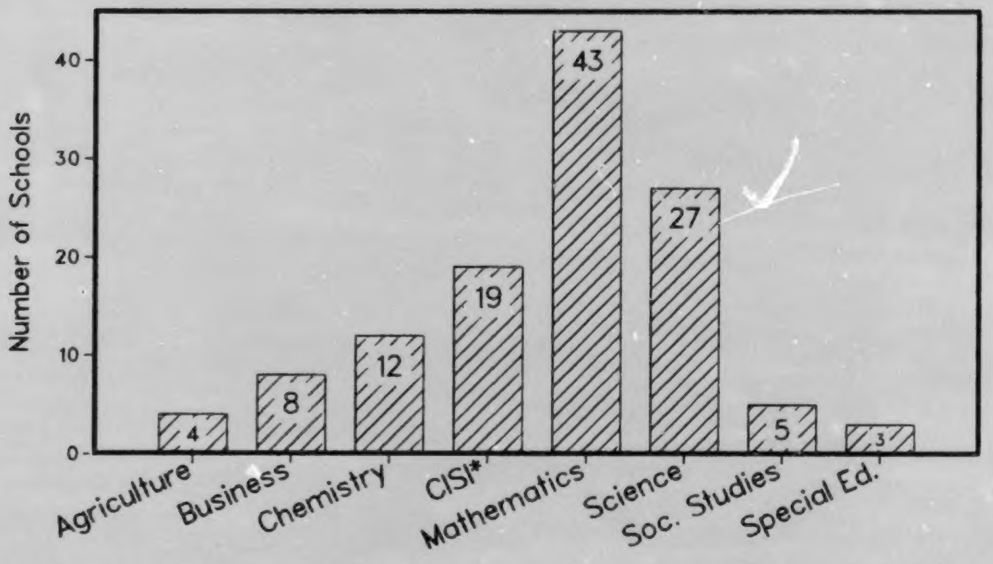
OBSERVATIONS

"Good Computing" existing now in a high school is not a function of the school's location—rural or urban, nor a function of the school's size. "Good computing" is rather a function of the activity of a teacher interested in computing and possessing the blessing of a sympathetic and encouraging administration. (A larger school with more teachers has a statistically better chance of having an interested teacher. Also a large city school district has the administrative capability of adjusting teacher assignments to provide interested teachers in each of its schools.)

I found high schools with under 70 students in which access to computing and instruction in programming is far superior to that available in high schools with over 700 students. I also found instances of administrators eagerly encouraging their teachers to use computers in the classroom with little result because of the lack of teacher interest; on the other hand, I found instances of teachers almost giving up because they cannot interest their administrators.

A "good computing" situation is well illustrated by a small school south of

Instances of Computer Use as a Tool



* Career Information System of Iowa

[†] Lest one fear that these data are badly skewed because of the University's presence in this region, a December 10 article in the Des Moines Register stated that 361 of 441 Iowa school districts, or 82%, have computing equipment.

Iowa City which is equipped with six microcomputers (3 Apple IIs and 3 TRS-80s) including the necessary peripherals and 2 terminals attached to a remote computer (HP3000). The "interested" teacher was teaching a full three semesters of programming and guiding projects for more advanced programmers. This teacher also was instrumental in encouraging teachers in most of the academic disciplines in his school to make use of the computers in their classes.

An "opposite extreme" situation is illustrated by an even smaller school in which there are two Apple II microcomputers installed in the rear of a mathematics classroom. The use of these computers is stymied by an administrative directive that allows use of the equipment only during the one hour per day that the computing language class of 12 students is taught in this classroom. Not only are the needs of the computing students unmet but a wide range of instructional use of these machines in other disciplines is eliminated.

Teacher preparation in most cases has been of the "bootstrap" variety—learning enough to stay ahead of the majority of the students, learning immensely from some of the brighter students. Many teachers have done a heroic job of taking on the responsibility of teaching computer programming at their school. They have attended workshops and spent many hours in study to bring what they considered an important new dimension into the student's education. The primary single omission in teacher preparation is in the design of programs. All but seven of the teachers I talked with have some training (formal or informal) in the language details of some computer language but only 4 have had a course in the methodology, style, and design of a program.

Thus, the design of programs has largely been omitted from the programming instruction with the expected result—programs that do the job but that are difficult to read, follow, and modify. These bad practices become habits which are hard to correct. My friend, Arthur Luehrmann, describes this best, as I paraphrase his statement—"The high school students cut their teeth on bad programming practices which require much orthodontia to correct in subsequent training."

There is another cause for concern—teacher preparation in good programming techniques and practices makes them extremely desirable in today's business and industrial world. Administrators worry about this and some teachers openly asked about such opportunities. Much effort must be put into solving the problem of retaining such teachers so that students may benefit from their expertise.

Textbooks have not been a large factor in the development of computing in the high schools. By and large they have served as inhibitors. This is primarily because the authors have not taken into account the needs of the teachers. Only a very limited number of texts have been tried by the teachers and most indicate that they are not at a level for student use. Some teachers complained that the textbook did not relate well to the microcomputers being used. The teachers are most comfortable with materials they or other teachers have developed. While this may produce some excellent materials, it has a high likelihood of degenerating into "The blind leading the blind."

Area Education Agencies have provided materials, particularly with respect to the Apple microcomputers. Many teachers use the Apple handbook and the Minnesota Educational Computing Consortium (MECC) materials (e.g., Introduction to Applesoft Basic). These are available at low cost under a contract negotiated by the State of Iowa with MECC.

Much effort is presently being concentrated on high school textbooks for computer programming. Some of these new texts are aimed at computer languages used on microcomputers. Hopefully, excellent texts with careful attention to design procedures will be available to meet this pressing need. Widespread marketing techniques will be necessary to get these better texts into the hands of the teachers.

Computing equipment acquisitions are expanding and will continue to expand; not only to meet the needs of the students in the computer programming classes but to allow students access to computers as tools of learning in all disciplines in the high school. Financial constraints prohibit meeting all needs immediately but most administrators expect yearly budgets to provide for continued increments in computing capabilities.

"Computing pressure" will increase on the high schools as more and more computing capabilities occur in the grade schools and junior highs. All but eleven of the 75 school districts indicated already present or planned computing activity in the grade schools or junior high schools. Students will expect to be able to use computers at least to the level available in earlier years. Add to this the present trend in home computers and we find an added impetus to increase use in the high school.

RECOMMENDATIONS

Computing equipment should be used on as wide a basis as possible. The best computing situations that I found during my visits were those in which the equipment had a focus but was accessible across the disciplines. The equipment should be located and regulated in such a way that students can have access from 7 am to 5 pm. It is such a valuable resource that, when feasible, it should be available not only to a specific class but across the student body for use in a variety of disciplines. Teachers should be allowed to take equipment home during holidays and weekends to become acquainted with a new program or to increase their skills.

It was evident during my visits that some teachers and administrators were jealously guarding the present equipment in order to provide certain students with enough computing power for their assignments. Although this is certainly necessary to some extent, a wider and more inclusive view of use of equipment will provide more support for the future acquisitions.

Some administrators feel that they have been "burned" before by audio-visual "messiahs" or individualized instruction prophets, and thus they are overly cautious. However, a microcomputer is an intellectual tool and a virtually necessary tool for teaching a necessary skill, computing, as well as a tool extremely useful in improving instruction in other disciplines. On the other hand, A-V equipment is none of these but merely a method of delivering instruction. To compare these is impossible!

Teacher training in computer programming should center on the design of good programs. Teacher training in the intricacies of a computer language is not sufficient to teach students how to produce good programs. Methodology, style, design steps, readability, linearity of control, and modularity need to be understood to produce well designed programs. Teaching students to write new programs to solve problems is the wrong emphasis when students who are hired into a programming position will almost entirely be revising previously written programs. Understanding this fact will change the emphasis from how to "go to", to be able to determine where the flow of the program "came from". It will also encourage the student to write well designed programs which are easily "readable" and "debuggable".

Most college and university computer science programs have undergone a corresponding change of emphasis over the past 7 or 8 years. The emphasis in early years was simply on teaching a language and its peculiar intricacies. Now most departments have moved to design of program emphasis, including structured programming and the principles of software engineering.

Teachers and prospective teachers, at all levels and in all disciplines, should take courses, attend workshops, and individually study to make themselves into computer literate teachers. This is important to begin a closing of the "computer gap" which is ever widening between students and some of their teachers.

Inservice teacher training in computer programming must take highest priority as a solution to the above stated problems. Although prospective science and mathematics teachers also need computer training, these prospective teachers will have almost no effect upon the high schools until the year 2000. This is because high school populations are shrinking or at best remaining stable and new teachers will rarely be hired by a declining school district. Therefore, special effort will have to be made to provide inservice institutes in the design of programs. These institutes should be aimed at a selected group of "good" teachers who expect to remain in teaching in spite of new skills learned.

PROJECTIONS

I anticipate that if most of the above recommendations are followed, we will see some of the following changes in Iowa and perhaps nationwide:

1. By 1985, the kindergarten through twelfth grade instruction will provide a student with a satisfactory level of computer literacy.
2. By 1990, most high schools will be teaching the first year of computing now being taught in colleges and universities. It is important that colleges and universities therefore be involved in the inservice teaching of design of programs.

CONCLUSION

My high school visits have provided new insights that clearly have implication for college and university computer science departments. The high schools are rapidly moving toward providing the beginning courses in computing. This will necessitate changes in computer science department curricula at the lower levels. We will undoubtedly need to develop valid testing devices to place students with varying levels of preparation.

To ensure some degree of uniformity in preparation and to avoid retraining in our courses, it is essential that computer science departments become intimately involved in the inservice training of present high school teachers.

REFERENCES

1. Luehrmann, Arthur, "Computer Literacy," EDUCOM Bulletin, Winter 1981, pp. 14, 15, 23.

2. Gleason, Gerald T., "Microcomputers in Education: The State of the Art," Educational Technology, March 81, pp. 7-18.

3. Poirot, Jim et al., "Teacher Education," TOPICS: (ACM-SIGCSE, SIGCUE), Jan. 81, pp. 18-26.

4. Myers, Darlene, "How Can Educators become Computer Literates?", The Computing Teacher, Vol. 8, No. 3 (1980,81), pp. 34-42.

Social Science

TEACHING SUBSTANTIVE SOCIOLOGY THROUGH COMPUTER-ASSISTED HOMEWORK: THE DEVELOPMENT OF CLEAR

George H. Conklin
Department of Sociology
North Carolina Central University
Durham, N.C. 27707
919-683-6222

INTRODUCTION

This paper describes the development of a new tutorial package for use by students in introductory sociology. The formal title of the new series of eight lessons is Computerized Learning and Review in Sociology or CLEAR for short. CLEAR is designed to be used on Apple microcomputers in conjunction with a printer or on mainframe computers in Basic with a printing terminal. Unlike previous packages for sociology such as PROFIS, CLEAR is designed to cover most units in introductory courses and is expected to be used with a specific textbook rather than with a variety of textbooks.

BACKGROUND

Sociology is a discipline where teaching and research are artifically separated even to a wider degree than is the case in most disciplines (Van Valey 1975). At the professional level, sociological research is conducted almost exclusively with data analysis which requires the use of the computer to make sense of large data sets. At the undergraduate level, however, few books emphasize the use of data even in advanced courses where one might reasonably assume pre-professional training takes place. Although for sociologists the computer is the main and even only laboratory tool, undergraduates are often trained for four years without being introduced to using a computer. As Stack (1979:211) has pointed out, the use of the computer in the classroom in sociology has "...been disproportionately in the area of data analysis and research methods. Many substantive areas of sociology have been underdeveloped."

Anderson (1981:188) has been able to locate only 22 computer-based packages for teaching all areas of sociology. As far as introduction to sociology is concerned, Anderson is able to list only three programs. One of the three programs for introductory sociology, GHETTO, is a computer version of a pencil and paper game, while TIPS is a sociological adaptation of a generalized testing procedure for use in all academic disciplines. The third available program is PROFIS, a set of tutorials covering three of about nine areas in most introductory sociology courses (see Conklin 1976) (1). Indeed, Anderson provides the data to prove the point raised by Stack, since instructors in sociology seem to feel that the use of computer-based instructional materials in sociology is most suitable for courses in research methods (89 percent), as opposed to theory where only 9 percent feel the use of the computer is appropriate. Introductory sociology, with 26 percent feeling computer-based instruction appropriate, falls between the two extremes. With only 9 percent in favor of using the computer for theory, one is left with the feeling that many sociologists are teaching theory which predicts events which instructors feel no obligation to look for in actual behavior!

In summarizing the future of the computer in instructional sociology, Anderson (1981:189) concludes, "The broad potential of the computer has not yet been realized in sociology. Many of the educational objectives which it supports have scarcely been demonstrated, not because they have been deemed irrelevant or exorbitant but because so few creative, instructional models have yet been constructed." Anderson thus finds that the lack of suitable software remains a barrier to the use of computer-assisted instruction in the field of sociology, especially in substantive areas. CLEAR was developed specifically to provide beginning sociology students with a series of tutorial lessons which would show that data and theory cannot be separated. CLEAR is designed to appeal to the introductory student and professor with the knowledge that few students enrolled in the class will become sociology majors. The emphasis for CLEAR was on developing lessons which would be inherently interesting to students, while at the same time, teaching principles of the social order which the liberal arts student ought to be aware of to be a functioning citizen in an ever-increasing technological setting.

THE SUBSTANTIVE AREAS

Computer-assisted instruction is not, at the present time, the sole basis of instruction in sociology, at least to the degree that it is possible to eliminate textbooks or traditional lectures. Development of computer-assisted sociology lessons is thus not an exercise which can be carried out in the abstract, without reference to current textbooks. For the development of PROFIS, the author had to assume that the lessons would be used ultimately with a wide number of different approaches to the study of society. For the development of CLEAR it became apparent that it might be better to try to tie the substantive point of each lesson to the specific discussion and approach of a specific textbook. In particular, it was the desire of the author to provide students with a coherent course where the computer-assisted element fits neatly with the printed medium, each providing its own apparent strengths.

But a problem arises when computer-assisted homework lessons such as CLEAR are tied to a particular book. The time an introductory book is in print without a revision can be as short as three years, while the amount of effort in a computer-assisted homework package makes the mean development time itself about three years. In order to get around this problem, major publishers were contacted through their field representatives. Each publisher was asked if it would supply the manuscript of a not-yet-published introductory sociology textbook. This would make it possible to write CLEAR to a specific text and yet have some guarantee that the book would be available for a reasonable amount of time before the first book revision might make CLEAR a somewhat less-than-excellent fit to the printed word.

Of the ten initial inquiries four publishers engaged in long conversations by phone. Of the four one eventually sent a manuscript which appeared to have excellent organization for use with CLEAR. The publisher's outline of the book (as yet untitled, but written by Henry Tischler and to be published by Holt, Rinehart and Winston) enabled CLEAR to be written so as to conform to the major sections of the new book. Table I presents the eight CLEAR lessons and how they fit into the organizational structure of the Tischler textbook. Note that the lessons cover the study of society, the individual and society, social organization and deviant behavior, social inequality, social institutions, and social change. In addition, there is an interesting appendix on the role of religion in determining deviant behavior. Several colleagues at North Carolina Central University cooperated with the author on three of the lessons, and their names are mentioned in the table.

THE DEVELOPMENT OF THE CLEAR LESSONS

The development of interesting computer-assisted tutorial homework assignments makes it necessary to pick topics which are inherently interesting in and of themselves. CLEAR thus strives to create an autotelic environment whereby the joys and frustrations of exploring the data and alternate logical possibilities are recreated for the user just as the thrill and excitement must have been felt by the original researcher (see Anderson 1981:172). Because the page proofs of the Tischler textbook were not ready, the goals of the lessons parallel the outline of the book but do not refer to specific pages or to specific examples of sociological practice given at the end of each chapter. Rather, CLEAR shows students how the ideas contained in the textbook may be applied to the study of society in a variety of contexts new to the student.

Technically CLEAR was written with transportability in mind. To the degree practical, use of string arrays was kept to a minimum while simple PRINT statments were used when possible. CLEAR1 introduces students to the way in which the CLEAR package asks for answers while having the class answer questions about themselves and their attitudes. CLEAR1 explains why answers were sometimes asked for as a 1 or a y (for yes), etc. To test for transportability, each lesson was passed without change from the original Apple version directly to an HP2000 minicomputer. After dimensioning for string responses such as yes or no, most CLEAR programs ran without further modification. Considerable thought was given to the use of video-only terminals, since some Apple computers may not be equipped with a printer. A video-only version of CLEAR is being produced, but the lessons are judged inferior to the printer versions by

Holt, Rinehart and Winston editors in terms of interest and student retention of the material. Given the fact that at the local Apple users group one of the hottest topics has been an extended discussion of the best printer for the Apple, it would seem that the severe limitations placed on the user by a video-only display are being recognized even by computer professionals. Since the CLEAR lessons have a homework assignment which is different for each student, a printer is much easier for the student than copying assignments from video screens.

Taken as a whole, the CLEAR series form not only a set of lessons but functions as a textbook. The lessons not only cover many of the traditional subjects in introductory sociology, but the printed output becomes the equivalent of a chapter in a textbook and is used by the authors as equivalent of an additional textbook chapter when examination time comes. Students are required to study CLEAR just as they would the regular textbook.

CLEAR2--AN EXAMPLE

Perhaps the best method of describing the CLEAR series is to study an example. CLEAR2 was the trial program submitted to the publisher for evaluation, in both a video version and a print version. The print version is reproduced in the appendix. The version reproduced here is from a teletype with a carbon ribbon, although we have found that Apple's thermal printer gives satisfactory output for students. The Decwriter II and the Epson MX80 have also been successfully tested and found satisfactory for use with the Apple.

CLEAR2 begins with Durkheim's definition of a social fact. Students are asked if they think physical objects such as a room are a social fact in addition to social norms. A series of alternate answers is given to point out that unless the student can walk on water, physical facts are indeed social facts too. However, unlike physical facts, social facts may be largely invisible in daily life. One of Durkheim's original uses of social facts was to explain differences in suicide death rates. CLEAR2 turns to modern mortality data to look at the mythical island of Melason. Actually Melason is Ceylon, now Sri Lanka and the data are, of course, real. However, by supplying the name of a mythical island students are encouraged to think for themselves and not rely on what they think might have happened.

The original funding for CLEAR came from the National Science Foundation (2) and emphasized that students in introductory sociology courses need to learn the math techniques used in sociology in conjunction with substantive sociology. Students in Part II of CLEAR2 ("Finding Social Facts") must master the idea of a death rate, dividing numbers and multiplying by 1000 to get a measure of deaths per thousand per year. But this is not all, for the idea is also developed that events take place which violate "common sense." On Melason DDT spraying may very well have "caused" a decline in the death rate after all because part of the island was not infected and thus not sprayed. The fact that the death rate declined in both parts of the island is introduced only toward the end of Part II of the lesson. How to set trend lines is difficult to introduce in just one lesson, so CLEAR2 gives students real data in a graph. Students can look at the real data in a graph from 1936 to 1953 which allows them to see how the data behaved over the years.

Under "Conclusions," the sociologically sound explanation is offered, giving the reasons listed by Fredriksen (1960, 1961 and 1966) as to what "caused" the decline in the death rate on Melason. Students are thus let in on the real issues after they have been fed limited information and allowed to make hasty conclusions. Students are told that the purpose of theory is to generate rival hypotheses which can thus be tested by data, not to make ideas conform to any one particular point of view. If sociology students need to learn any one thing, it is "be skeptical." The goal of CLEAR2, to integrate both methods and theory, is accomplished by telling the student that theory leads to fact which then leads to formulation of new theory.

The "homework" section of CLEAR2 gives a student one of 100 stored homework assignments. While the HP version will access with equal probability all 100 assignments, the Apple version would give the same assignment each time the the disk is booted. For this reason CLEAR is set to run the random number generator when answers are given throughout the lesson. Unless all answers are the same each time and unless no errors are made, assignments will be "randomly" given. Students must use the printed worksheet to record their calculations, and record their responses in ink. Fraternity files or the modern equivalent would be made somewhat awkward by having so many different assignments available.

THE INSTRUCTOR AND CLEAR

Gelles (1980:16) has demonstrated that innovation in the institution of higher education does not outlast the innovator, at least in sociology courses. One problem with the use of computer-assisted instruction with homework assignments given at random is that an additional load is put upon the instructor to correct a huge pile of extra papers. Built into CLEAR, therefore, is a system of programs so student may self-correct their assignments through the use of a "correction code." The faculty asks that the assignments be turned in with the homework done in ink. The assignments are simply checked in and returned to the student. The next week the students put their correction codes into CLEAR and obtain the answers. Students thus are able to correct their own work, see their errors, and learn from their errors. The instructor need not be concerned with the process unless the student asks for help. We currently collect all CLEAR corrections before the examination to see how the students are doing. Writing in ink discourgages students from erasing answers to make it appear that they have a perfect score. If there are many errors students are simply asked to do the assignment over.

Lessons are provided for eight weeks, or roughtly every other week of an introductory sociology course. The every-other-week schedule seems to fit a natural rhythm for sociology courses. It is recommended that CLEAR be used under the concept of competency-based education, where homework is provided for the student as practice and not a graded exercise. Even though homework is not graded and this fact is explained to students, the old high school math model of grading homework remains in the students' mind. Thus the only real problem found with the actual use of CLEAR2 was that some students would run the assignment over so the line telling the student how many "errors" they made would say that they made none at all. We see what is learned by the student by putting the material from the homework assignments on the exam, thus removing pressure from the student to have a perfect computer run. By letting CLEAR be "student run," we also hope to remove any additional burden of CLEAR from the faculty work load. For this reason we have made no effort to have the computer keep track of student responses or "scores," since such a system not only adds to the faculty work load, it also detracts from the desired autotelic element built into CLEAR.

CONCLUSIONS

The design of CLEAR was made on the assumpiton that it is not technically interesting programming which makes a good tutorial, but that a good intellectual issue is what creates interest. CLEAR was also designed with the idea that instructors should not be burdened with increased work due to using the computer, while students ought to be given individualized homework assignments. Yet the issue does need to be addressed as to what might be gained by using Apple's high resolution color graphics facility and the ability of microcomputers in general to display charts, tables and graphs more clearly than the usual printer plot. In the case of CLEAR2, the map of Melason can be made quite spectacular with the malaria areas in red and the "safe" area in green. The effect is much better than just using X's and O's. Further, the semi-log plot can become quite detailed using high resolution graphics, close to the printed table presented by the original author. But color cannot be used on the printers, while even Apple's thermal printer does not do very well with graphics. From a teaching point of view, little is gained by using more sophisticated displays, while the time required to program each lesson rises greatly and transportability between machines ceases to exist. We designed CLEAR to be transportable and feel that the compromises to meet this goal have not been serious.

CLEAR1 to 8 cover all areas of introductory sociology. As such, CLEAR is designed to meet the needs stated by Stack, Anderson, and others for a series of computer-assisted assignments to cover substantive areas of sociology. CLEAR will be distributed to professors by Holt, Rinehart and Winston when the new Tischler textbook "Introduction to Sociology" (tentative title) is announced. Readers who wish to try out CLEAR2 may request an HP2000 listing in Basic from the author.

FOOTNOTES

(1) PROFIS, now distributed by CONDUIT, is being revised by Conklin for use on the Apple. The lessons are being enlarged and improved and use Apple high resolution graphs.

(2) ThE CLEAR series was developed under a Minority Institution Science Improvement Special Project Grant #SER8017631. Equipment to run the programs was provided in part by a NSF grant from Instructional Scientific Equipment Program for Undergraduate Science Education #SER8015418. Any opinions, conclusions, or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE I
AN OUTLINE OF THE CLEAR TUTORIAL SYSTEM

1) CLEAR1. CLEAR1 is the equivalent of the Preface to the textbook. CLEAR1 explains why sociologists use data and tells the student how to answer questions in the CLEAR series. The questions themselves let the instructor build a demographic portrait of the class.

2) CLEAR2. This lesson is included in the appendix. It is used with Part I of the textbook, "The Study of Society," and ties together the chapters on "The

Sociological Perspective" and "Doing Sociology:Research Methods."

3) CLEAR3 covers Part II of the textbook, "The Individual in Society." CLEAR3 is entitled "Beliefs" and discusses how people will make and follow decisions based on what they believe is true, not on what is "factually" true. Students rank the nations' jobs by how "safe" they are, and learn about rank order correlations in checking their beliefs with social facts.

4) CLEAR4 covers "The Social Organization of Society," which takes in deviant behavior and urban society. CLEAR4 lets students explore a data set which is supposed to show that the moon has some influence on the murder rate. Students learn how to take averages to iron out experimental error and to be skeptical of data.

5) CLEAR5 is a discussion of social inequality, and covers a unit in the textbook on social stratification. Written by Isaac Robinson at North Carlina Central University, CLEAR5 is subtitled "money" and shows how a college education may not pay any more. Students learn to project the past into the future using a series of trend lines showing that the traditionally expected increase in the standard of living in the United States is a fast-fading dream.

6) CLEAR6 is about "Social Institutions," marriage specifically. Students arme given a real marriage predictor test, which is then graded by the computer and interpreted by a series of 10 resident experts who spew out strong ideology on the use and misuse of data in predicting behavior. Students learn how to score tests and how many standardized tests have one question which tends to overload the other items. This lesson is particulary important for the liberal arts student who may be later asked to place faith in standardized tests.

7) CLEAR7 is a simulation on "Social Change," covering a section of the textbook by that name. Using the concept of relative deprivation, students are asked to build a portrait of a typical black militant (Militant-X). Students then test their ideas against real data by use of tables using a control group. Dependent and independent variables are the issue, a topic recently covered on the National Teacher Exam (general knowledge section). This lesson was written by Isaac Robinson.

8) CLEAR8 is about "Religion" and is included as the appendix to the series. CLEAR8, by Frank Scherer at North Carolina Central University, compares data on church attendance with the national crime rate. The results surprise most students.

REFERENCES

Anderson, R.E. (1981) 'Instructional computing in sociology:current status and future prospects.' "Teaching Sociology" (January):171-196.

Conklin, G.H. (1976) 'Using the computer in introductory sociology.' "Teaching Sociology" (October):83-95. Note: this article discusses the original PROFIS.

Frederiksen, H. (1960) 'Malaria control and population pressure in Ceylon.' "Public Health Reports" (October):865-868.

- - - (1961) 'Determinants and consequences of mortality trends in Ceylon.' "Public Health Reports" (August):659-663. This article contains the semi-log plot used in CLEAR2.

- - - (1966) 'Determinants and consequences of mortality and fertility trends.' "Public Health Reports" (August):715-727. This article generalizes the argument from Ceylon to the world as a whole.

Gelles, R.J. (1980) '"Teaching Sociology" on teaching sociology (sic).' "Teaching Sociology" (October):3-20.

Stack, S. (1979) 'Teaching demography through the computer.' In "Proceedings of the NECC 1979 National Educational Computing Conference," Dianna Harris, editor. University of Iowa:211-219.

Van Valey, T. (1975) 'Methods or not: an examination of introductory texts.' "Teaching Sociology" (October):21-32.

```
                          CLEAR2


    THERE EXIST IN SOCIETY RELATIONSHIPS BETWEEN THE WAY THE SOCIETY
IS ORGANIZED AS A WHOLE AND THE WELL-BEING OF THE INDIVIDUAL.  EMILE
DURKHEIM (1858-1917), A FRENCH SOCIOLOGIST, STUDIED SUICIDE AND FOUND
THAT THE LESS COHESIVE A GROUP OF PEOPLE WERE, THE MORE LIKELY INDI-
VIDUALS IN THE GROUP WERE TO KILL THEMSELVES.  FOR EXAMPLE, PROTESTANTS
WERE MORE LIKELY TO COMMIT SUICIDE THAN CATHOLICS.

    WHAT WAS DURKHEIM'S EVIDENCE?  DURKHEIM COLLECTED DATA ON SUICIDE
AND CALCULATED DEATH RATES TO MAKE HIS CONCLUSIONS.  HE CONSIDERED
SUICIDE DEATHS TO BE 'SOCIAL FACTS' AS WELL AS PHYSICAL FACTS.  DURKHEIM
DEFINED 'SOCIAL FACTS' AS WAYS OF ACTING, THINKING AND BELIEVING WHICH
ARE EXTERNAL TO THE INDIVIDUAL AND ENDOWED WITH THE POWER OF COERCION.

    SOCIAL FACTS THUS HAVE POWER OVER YOU.

    CAN A BUILDING BE A SOCIAL FACT?  TYPE 1 FOR YES OR 2 FOR NO
?2
    THINK FOR A MINUTE.  DOES NOT A WALL FORCE YOU TO CHANNEL YOUR
BEHAVIOR IN FIXED WAYS?

    CAN A BUILDING BE A SOCIAL FACT?  TYPE 1 FOR YES OR 2 FOR NO
?2
    UNLESS YOU CAN WALK ON WATER, PHYSICAL BARRIERS INDEED CONSTRAIN
YOU.  IF YOU CAN WALK ON WATER, YOU ARE AN EXCEPTION TO THIS RULE.
    ROOMS DO PATTERN OUR SOCIAL INTERACTIONS.  ONCE BUILT, BUILDINGS
CONTINUE TO REGULATE BEHAVIOR, OFTEN IN WAYS TOTALLY UNEXPECTED BY
ARCHITECTS.

    LET US CONTINUE.

    IS A LAW AGAINST DRIVING THROUGH A RED LIGHT A 'SOCIAL FACT'?
TYPE 1 FOR YES OR 2 FOR NO AND PUSH RETURN.
?2
    DOES NOT A LAW HAVE SOME INFLUENCE?

    IS A LAW AGAINST DRIVING THROUGH A RED LIGHT A 'SOCIAL FACT'?
TYPE 1 FOR YES OR 2 FOR NO AND PUSH RETURN.
?2
    COME OFF IT!  IF YOU BREAK THE LAW ENOUGH, THE ENFORCERS OF THE
SOCIAL FACTS KNOWN AS POLICE WILL PUT YOU IN ANOTHER SOCIAL FACT KNOWN
AS JAIL, ALSO A PHYSICAL FACT, IN CASE YOU WERE WONDERING.

    SOCIAL FACTS ARE NOT ONLY PHYSICAL CONSTRAINTS, BUT SOCIAL AND
LEGAL LAWS (NORMS) ARE TOO.  LAWS ARE PUBLISHED SO WE ALL, IN THEORY,
CAN READ THEM AND OBEY.

    ARE ALL SOCIAL FACTS APPARENT TO YOU AS A PUBLISHED LAW MIGHT BE?
CAN A 'SOCIAL FACT' INFLUENCE BEHAVIOR EVEN WHILE THE INDIVIDUAL BEING
CONTROLLED IS UNAWARE OF THE INFLUENCE OF THE SOCIAL FACT?

IF YOU THINK YES, ENTER 1.  IF YOU THINK NO, ENTER 2.
?2

    WELL, SOCIAL FACTS INDEED CONTROL BEHAVIOR EVEN IF YOU ARE UNAWARE
OF THEM.  SOME GROUPS OF PEOPLE HAVE A HIGHER OR LOWER BIRTH RATE, DEATH
RATE OR EVEN SUICIDE RATE THAN OTHERS, YET THE INDIVIDUAL IN SOCIETY IS
PROBABLY UNAWARE OF THE DIFFERENCES.


                PART II FINDING SOCIAL FACTS

    DURKHEIM LOOKED FOR SOCIAL FACTS IN THE RECORDS PEOPLE LEFT BEHIND
OF THEIR ACTIONS.  LET US CONSIDER NOT ONLY DEATH FROM SUICIDE, BUT
DEATH FROM ALL CAUSES ON A TROPICAL ISLAND WE WILL CALL MELASON.
    THE MAP OF MELASON IS AS FOLLOWS.  THE ISLAND IS SURROUNDED BY WATER
```

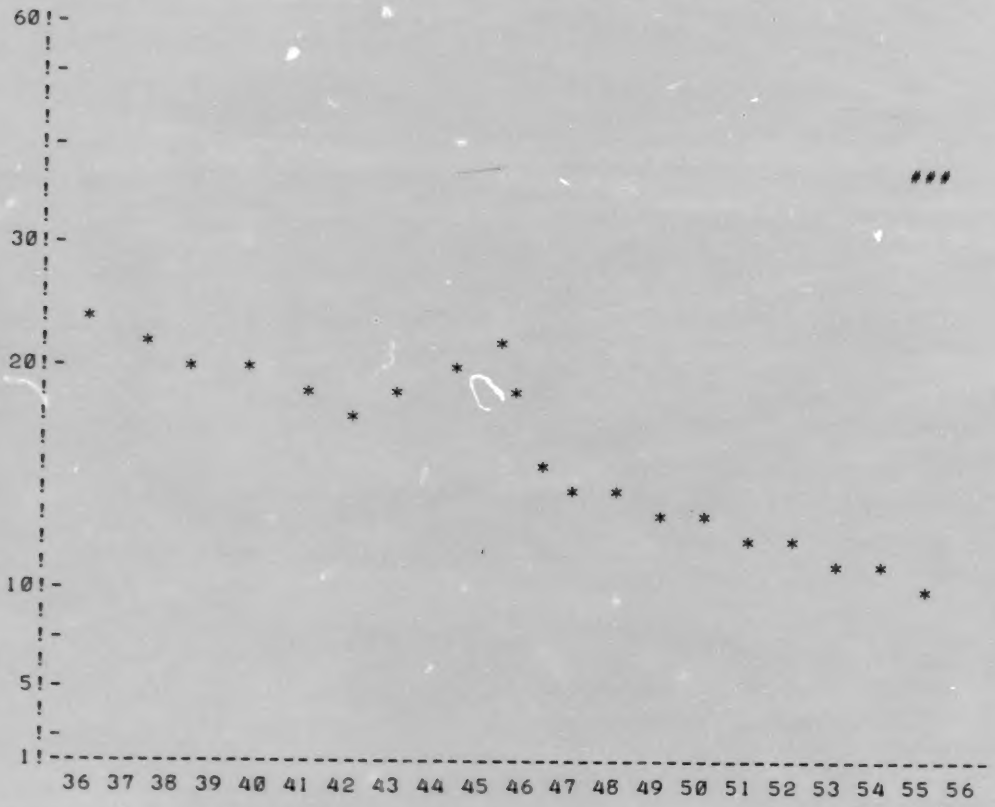
```
          MAP OF THE ISLAND


              ***
               ***
               *****
                **OOOO
              OOOOOOOOO
              OOOOOOOOOO
              OOOOOOOOOOO
             OOOOOOOOOOOOO
            OOOOOOOOOOOOOOO
  OCEAN    OOOOOOOOOOOOOOOOO
            OOOOOOOOOOOOOOOO
            OOOOOOOOOOOOOOOOO     (MANY MILES TO NEAREST LAND)
             OOC*****OOOOOOOOO
            **********OOOOOOO
             ********OOOOOOOO
(NO OTHER    ****OOOOOOOOOOO
ISLANDS      ****OOOOOOOOOO
NEAR)        ****OOOOOOOOO
             ****OOOOOOOO
              ***OOOOO
               **OOO
                *OO

     *=REGION 1.   O=REGION 2.


     BECAUSE OUR SOCIETY IS LOCATED ON AN ISLAND, IT IS POSSIBLE TO
INTRODUCE CHANGES QUITE QUICKLY WHICH MIGHT AFFECT THE QUALITY OF LIFE.
INDEED, THIS HAPPENED.  BETWEEN 1946 AND 1947 MELASON'S DEATH RATE
FELL SHARPLY.  CONSIDER THE FACTS PLEASE:

          POPULATION IN       CRUDE DEATH RATE        # OF DEATHS IN ONE
  YEAR       MILLIONS         (PER 1,000 PEOPLE)      YEAR IN 1,000'S
---------!-----------------!-----------------------!--------------------
  1938          5.8                 21.0              122.8 OR 122,800
---------!-----------------!-----------------------!--------------------
  1940          5.9                 20.6                   121.5
---------!-----------------!-----------------------!--------------------
  1943          6.3                 ????                   131.0
---------!-----------------!-----------------------!--------------------
  1944          6.4                 20.8                   133.1
---------!-----------------!-----------------------!--------------------
  1945          6.4                 21.5                   138.6
---------!-----------------!-----------------------!--------------------
  1946          6.5                 19.8                   128.7
---------!-----------------!-----------------------!--------------------
  1947          7.0                 14.0                    98.0
---------!-----------------!-----------------------!--------------------
  1948          7.2                 13.2                    95.0
---------!-----------------!-----------------------!--------------------
  1953          8.3                 10.9                    90.5
---------!-----------------!-----------------------!--------------------

WHAT WAS THE DEATH RATE PER 1,000 PERSONS IN 1938?
ENTER THE NUMBER FROM THE TABLE AND PUSH RETURN.
?1
WHAT WAS THE DEATH RATE PER 1,000 PERSONS IN 1938?
ENTER THE NUMBER FROM THE TABLE AND PUSH RETURN.
?1
WHAT WAS THE DEATH RATE PER 1,000 PERSONS IN 1938?
ENTER THE NUMBER FROM THE TABLE AND PUSH RETURN.
?1
USE 21.0 AND PUSH RETURN.
WHAT WAS THE DEATH RATE PER 1,000 PERSONS IN 1938?
ENTER THE NUMBER FROM THE TABLE AND PUSH RETURN.
?
```

```
 21.0

WHAT WAS THE DEATH RATE IN 1947, THE YEAR AFTER THE GREAT DROP?
ENTER THE NUMBER AND PUSH RETURN.
?14

    COMPARED TO AS LATE AS 1945, THIS REPRESENTS A DROP OF ABOUT 35
PERCENT!!  THIS DROP HAPPENED SUDDENLY TOO.  IS THERE A MATH ERROR?

    THE CRUDE DEATH RATE IS DEFINED AS FOLLOWS:

                          # OF DEATHS IN A YEAR
CRUDE DEATH RATE = ----------------------- * 1000
                     POPULATION IN THAT YEAR

    LET US CALCULATE THE DEATH RATE IN 1938.  HOW MANY PEOPLE, ACCORD-
ING TO THE TABLE, DIED ON MELASON IN 1938?  ENTER THE NUMBER WITH THE
RIGHT NUMBER OF ZEROS (AND NO COMMAS) AND PUSH RETURN.
?122800
O.K., NOW HOW MANY PEOPLE WERE ON THE ISLAND IN 1938?  ENTER THE NUMBER
IN MILLIONS USING THE DECIMAL AND PUSH RETURN.
?1
TRY 5.8 (POPULATION IN MILLIONS).
O.K., NOW HOW MANY PEOPLE WERE ON THE ISLAND IN 1938?  ENTER THE NUMBER
IN MILLIONS USING THE DECIMAL AND PUSH RETURN.
?5.8
YOUR FORMULA IS THUS:
                          122800
                         -------- * 1000 WHICH EQUALS 21.0
                         5800000
THIS DOES SEEM TO BE THE SAME AS THE TABLE.

WELL, IT LOOKS AS IF THE NUMBERS ARE RIGHT FOR 1938.  LET US TRY 1943.
HOW MANY DEATHS WERE THERE IN 1943?  ENTER THE NUMBER FROM THE TABLE
USING THE PROPER NUMBER OF ZEROES AND NO DECIMALS OR COMMAS.
?1
YOUR RESPONSE IS INCORRECT, TRY AGAIN.
HOW MANY DEATHS WERE THERE IN 1943?  ENTER THE NUMBER FROM THE TABLE
USING THE PROPER NUMBER OF ZEROES AND NO DECIMALS OR COMMAS.
?1
USE 131000
HOW MANY DEATHS WERE THERE IN 1943?  ENTER THE NUMBER FROM THE TABLE
USING THE PROPER NUMBER OF ZEROES AND NO DECIMALS OR COMMAS.
?131000
NOW ENTER THE NUMBER OF PEOPLE ON MELASON IN 1943 IN MILLIONS, USING
THE DECIMAL.
?6.3
THE FORMULA IS NOW COMPLETE:
                          131000
                         -------- * 1000 WHICH EQUALS 20.8
                         6300000

TAKE YOUR PENCIL AND FILL IN 20.8 IN THE SPACE PROVIDED ABOVE.  WHEN
YOU ARE DONE, TYPE 1 AND A RETURN TO CONTINUE.
?1

    WELL, THE DATA DO SEEM TO BE CALCULATED CORRECTLY.  WHAT HAPPENED?
BELOW ARE LISTED SOME POSSIBILITIES.  PICK THE ONE YOU FEEL MOST LIKELY.

POSSIBILITY #                    OPTION
     1          PENICILLIN WAS INTRODUCED BY THE U.S. ARMY, SAVING
                THOUSANDS OF LIVES.

     2          THE GOVERNMENT OF MELASON SOCIALIZED MEDICINE.

     3          THE WATER SUPPLY WAS CLEANED UP.
     4          DDT WAS SPRAYED, CONTROLLING MALARIA ON MELASON, A
                TROPICAL ISLAND SURROUNDED BY OCEAN.

     5          THERE WAS AN END TO COLONIAL RULE, ALLOWING ALL TO
                BE TREATED EQUALLY.

ENTER ONE OF THE POSSIBILITIES (1 TO 5) AND PUSH RETURN.
?1
THIS IN FACT DID NOT HAPPEN.  TRY ANOTHER OPTION.

ENTER ONE OF THE POSSIBILITIES (1 TO 5) AND PUSH RETURN.
?5
COLONIAL RULE CONTINUED.  TRY ANOTHER OPTION.

ENTER ONE OF THE POSSIBILITIES (1 TO 5) AND PUSH RETURN.
?2
WHILE INDUSTRIALIZED NATIONS USUALLY WENT THIS ROUTE, THE ISLAND OF
MELASON DID NOT.  TRY ANOTHER OPTION.

ENTER ONE OF THE POSSIBILITIES (1 TO 5) AND PUSH RETURN.
?3
CLEAN WATER IS IMPORTANT, BUT NOT THE KEY HERE.  TRY AGAIN.

ENTER ONE OF THE POSSIBILITIES (1 TO 5) AND PUSH RETURN.
?4
RIGHT!!!!!!!!!!!
     DDT SPRAYING WAS BEGUN IN 1945 AND WENT ON SEVERAL YEARS.  THE
DROP IN THE DEATH RATE OVER THE FIRST TWO YEARS WAS QUICK AND EASY TO
SEE.  NOTICE THAT THE NUMBER OF DEATHS, WHICH HAD BEEN INCREASING
YEARLY, SUDDENLY BEGAN TO FALL.  SWOLEN SPLEENS, A PHYSICAL SIGN OF
MALARIA, WERE HAD BY 50 TO 75% OF THE POPULATION BEFORE SPRAYING.
MOSQUITOS CARRY MALARIA AND DDT KILLED MOSQUITOS.  THE CHEMICAL COM-
PANIES CALLED IT BETTER LIVING DUE TO CHEMISTRY.
      IT SOUNDS LIKE AN OPEN AND SHUT CASE, RIGHT?  HOW CONVINCED ARE YOU

     WHAT ABOUT RIVIAL HYPOTHESES?  DO YOU THINK WE NEED TO LOOK FURTHER?

 1    NO, THE EVIDENCE IS STRONG
 2    NOT SURE
 3    WE SHOULD LOOK FURTHER

ENTER 1,2 OR 3 AS YOU BELEIVE AND PUSH RETURN.
?1

     REALLY?  IS THAT SO!  IT JUST SO HAPPENS THAT WE MUST LOOK FURTHER
BECAUSE THE ISLAND OF MELASON IS DIVIDED INTO TWO REGIONS.  THE MAP
SHOWS THAT REGION 1 (*) WAS SMALLER THAN REGION 2 (O).  IN REGION 2
MALARIA WAS EVER-PRESENT.  BUT IN REGION 1, THERE WAS VERY LITTLE.

     PROBLEM:  IF THE MALARIAL PART OF MELASON SHOWED A SHARP DECLINE
IN THE DEATH RATE (AS IT DID), WHAT SHOULD THERE HAVE BEEN IN REGION 1?
WHAT DO YOU THINK?
 1    THERE WAS NO DROP IN REGION 1, SINCE IT WAS NOT SPRAYED
 2    THERE WAS A DROP OF ABOUT 1-9% IN REGION 1, DUE TO A 'CARRY
      OVER' EFFECT
 3    THERE WAS THE SAME DROP OF ABOUT A THIRD IN BOTH REGIONS.

ENTER YOUR ANSWER (A NUMBER FROM 1 TO 3 AND PUSH RETURN).
?
```

```
THE DATA ARE AS FOLLOWS.

          DEATH RATES ON MELASON 1944-1948
          AREA SPRAYED  AREA NOT SPRAYED
                         (CONTROL GROUP)
---------!------------!----------------
  1944         21.0          19.2
---------!------------!----------------
  1945         23.8          19.1
---------!------------!----------------
  1946         17.6          14.5
---------!------------!----------------
  1947         14.6          13.7
---------!------------!----------------
  1948         13.6          14.1
---------!------------!----------------

DO YOU WISH TO CHANGE YOUR ANSWER NOW, HAVING SEEN THE FACTS?
ENTER  1  YES I WISH TO CHANGE MY ANSWER
ENTER  2  NO  I'LL STICK WITH IT

?1
   TOO LATE.  YOU CANNOT TRY TO CHANGE YOUR MIND AND MESS
WITH MY MEMORY SYSTEM.

    HOWEVER, THE DEATH RATES DROPPED ALMOST EQUALLY IN BOTH AREAS OF
MELASON. YOUR ANSWER OF 2 IS THUS NOT ACCURATE.

SOME FACTOR OTHER THAN DDT MUST BE RESPONSIBLE BECAUSE OUR CONTROL
GROUP SHOWED THE SAME DROP.

    WHAT ABOUT GOVERNMENT EXPENDITURES FOR HEALTH?  DID SUCH EXPENDI-
TURES RISE BETWEEN 1938 AND 1947?
     1      THEY MUST HAVE RISEN
     2      NO IDEA
     3      THEY FELL

INPUT YOUR ANSWER (1 2 OR 3 AND PUSH RETURN).

?1
    THEY DID NOT RISE.  GOVERNMENT EXPENDITURES FOR HEALTH ACT-
UALLY FELL A LITTLE.

LET US ASK YOU AGAIN THE QUESTION WHICH WE DID EARLIER.
WHAT ABOUT LOOKING FURTHER STILL FOR ANSWERS.  DO YOU THINK WE NEED
TO LOOK AT MORE FACTS?
  1  NO, THE EVIDENCE IS STRONG ABOUT DDT
  2  NOT SURE
  3  WE SHOULD LOOK FURTHER
ENTER 1, 2 OR 3 AND PUSH RETURN.
?
```

```
    HERE IS A TREND LINE SHOWING THE DEATH RATE ON MELASON FROM 1936
TO 1942.

60!-
  !
  !-
  !
  !
  !-
  !
  !
  !
30!-
  !
  !
  !  *
  !     *
20!-       *  *
  !              *  *
  !                    *
  !
  !
  !
  !
  !
  !
10!-
  !
  !-
  !
 5!-
  !
  !-
 1!---------------------------------------------------------------
    36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56

    USING A STRAIGHT LINE SUCH AS THE EDGE OF A PIECE OF PAPER OR
LETTING YOUR EYE PROJECT THE LINE SHOWN, WHAT WOULD BE THE PROJECTED
DEATH RATE IN 1955, ASSUMING THAT THE TREND FROM 1936 TO 1942 CONTINUED
UNCHANGED?
    YOU NEED TO READ YOUR ANSWER USING THE SCALE AT THE LEFT OF THE
GRAPH.
ENTER YOUR PROJECTED DEATH RATE FOR 1955 (FROM 5 TO 60) AND PUSH RETURN.
?45
YOUR VALUE IS TOO HIGHER.  LOWER IT AND TRY AGAIN.
    YOU NEED TO READ YOUR ANSWER USING THE SCALE AT THE LEFT OF THE
GRAPH.
ENTER YOUR PROJECTED DEATH RATE FOR 1955 (FROM 5 TO 60) AND PUSH RETURN.
?1
YOUR VALUE IS TOO LOW. RAISE IT AND TRY AGAIN.
    YOU NEED TO READ YOUR ANSWER USING THE SCALE AT THE LEFT OF THE
GRAPH.
ENTER YOUR PROJECTED DEATH RATE FOR 1955 (FROM 5 TO 60) AND PUSH RETURN.
?11
GOOD PROJECTION!!
```

LONG TERM TRENDS ON MELASON WERE THUS RELATIVELY CONSTANT OVER QUITE A PERIOD OF TIME. DDT SPRAYING IS THUS NOT THE ONLY 'CAUSATIVE'.



WHAT WAS CHANGING ON MELASON? THE 'SOCIAL FACT' THAT WAS CHANGING WAS THAT THE FOOD SUPPLY WAS GETTING BETTER AND BETTER, HAVING EXPERIENCED A BRIEF INTERRUPTION DURING WORLD WAR II. CONSIDER THE AVERAGE WEIGHTS OF SCHOOL CHILDREN:

AVERAGE WEIGHT IN POUNDS

	BOYS		GIRLS	
AGE	1945	1950	1945	1950
6	35	38	34	37
10	48	52	48	52
12	53	59	56	63
14	64	75	70	79

IN SHORT, EVERYONE WAS BETTER FED. WHY? SOCIAL POLICY CHANGED, AND THE ECONOMY ADVANCED.

SINCE YOUR PROJECTION WAS OFF, THE ### ON THE TREND LINE SHOWS YOU HOW FAR OFF YOU WERE, IN A GENERAL SORT OF WAY.

CONCLUSIONS
USE OF SOCIAL FACTS.

IN YOUR OPINION, THE NEED FOR RIVAL HYPOTHESIS WAS AS FOLLOWS:
1. FIRST TIME ASKED 'NO, THE EVIDENCE IS STRONG'
2. SECOND TIME ASKED 'NOT SURE'

OBVIOUSLY, YOU HAVE BEEN INFLUENCED BY THE DATA.

THERE IS, IN ANY CASE, ALWAYS A NEED TO LOOK FOR RIVAL HYPOTHESES.

```
    OUR ISLAND ACTUALLY EXISTS.  IT IS SRI LANKA, FORMERLY CEYLON.
THE DATA YOU HAVE USED AND SEEN ARE REAL.  WHAT CHANGED ON SRI LANKA?

    FIRST, GENERAL ECONOMIC CONDITIONS IMPROVED.  THIS CAUSED HEALTH
TO IMPROVE, DESPITE THE GENERAL LACK OF DOCTORS.

    SECONDLY, IF DURKHEIM WERE ALIVE TODAY, HE MIGHT POINT OUT THAT
PUBLIC POLICY AS MUCH AS INDIVIDUAL ACTION IS THE 'SOCIAL FACT' WHICH
INFLUENCES HOW LONG YOU MAY LIVE.  TRYING TO STAY 'HEALTHY' IN A
SOCIETY WITH MANY 'SICK' PEOPLE IS DIFFICULT.  ONCE THINGS BEGAN TO
IMPROVE, THERE WAS A CERTAIN MOMENTUM.

    FURTHER, CEYLON, UNTIL RECENTLY, ACTUALLY SUBSIDIZED RICE SO
EVERYONE, RICH OR POOR, HAD ENOUGH TO EAT.  IT WAS THEIR VERSION
OF FOOD STAMPS, AVAILABLE TO ALL.  EVEN WITHOUT MANY DOCTORS, SRI
LANKA BECAME A HEALTHY NATION.  TODAY, SRI LANKA'S DEATH RATE COMPARES
VERY WELL WITH THAT IN THE INDUSTRIAL NATIONS OF THE WORLD, EVEN IF
ON THE SURFACE, PEOPLE HAVE FEWER PHYSICAL POSSESSIONS.

    IN CONCLUSION, ALWAYS LOOK FOR RIVAL HYPOTHESES TO EXPLAIN 'SOCIAL
FACTS'.  EVERYONE, INCLUDING YOUR GRANDMOTHER, HAS AN OPINION ABOUT
SOCIETY.  BUT EVERYONE'S GRANDMOTHER DOES NOT AGREE.  RIGHT?

    DURKHEIM USED DEATH RATES FROM SUICIDE TO EXPLAIN HIS LOGIC.  WE
HAVE USED DATA FROM A WHOLE NATION.  BOTH DURKHEIM AND YOU HAVE USED
'SOCIAL FACTS' IN EXPLORING SOCIAL CHANGE.

                         MISCELLANEOUS FACTS:
  1.  DID YOU KNOW THAT DURKHEIM COMMITTED SUICIDE IN 1917?
  2.  YOU MADE 6 ERRORS ON THE DATA INTERPRETATION SECTION.

                               HOMEWORK
                         COMPUTING SOCIAL FACTS

    FOR HOMEWORK, CALCULATE THE FOLLOWING 10 DEATH RATES FROM THE DATA
PRESENTED.  NEXT WEEK YOU CAN PUT YOUR 'CORRECTION CODE' INTO THE
COMPUTER AND CORRECT YOUR ANSWERS TO SEE IF YOU UNDERSTAND THE PROCESS.

YOUR CORRECTION CODE FOR NEXT WEEK IS 88

                                                DEATH RATE
EXAMPLE #   # OF DEATHS   TOTAL POPULATION   PER 1,000 (FILL IN)
----------!-------------!-------------------!--------------------
    1          49414            3394492
----------!-------------!-------------------!--------------------
    2          52676            8313184
----------!-------------!-------------------!--------------------
    3          56602            8309662
----------!-------------!-------------------!--------------------
    4          80294            4457786
----------!-------------!-------------------!--------------------
    5          13379            2646793             5.05
----------!-------------!-------------------!--------------------
    6          86418            6985572
----------!-------------!-------------------!--------------------
    7          37024            3941532
----------!-------------!-------------------!--------------------
    8          67948            5801748
----------!-------------!-------------------!--------------------
    9          37665            1202323
----------!-------------!-------------------!--------------------
   10          67407            9431362
----------!-------------!-------------------!--------------------


    TURN IN YOUR ANSWERS IN INK TO YOUR INSTRUCTOR.  REMEMBER TO
NOTE YOUR CORRECTION CODE.  YOU WILL NEED IT NEXT WEEK TO CORRECT
YOUR ANSWERS.

DONE
```

STATAN, A QUERY LANGUAGE
TO HELP STUDENTS PRACTICE STATISTICS

Anne M. Parkhurst
Biometrics and Information Systems Center
103 Miller Hall
University of Nebraska
Lincoln, Nebraska 68583-0712
(402) 472-2903

I. INTRODUCTION

Computers are very important in the study of statistics. Computers are used to simulate models, generate distributions, and illustrate concepts. But by far the most preponderant use of computers is to produce results of statistical procedures. Several nationally-known statistical packages have been written for this purpose. However, from the student's point of view, the major drawback to the use of these packages is the need to learn the control language for each system. A frequently voiced complaint is "This is supposed to be a course in statistics not computer science." This is especially true of the service-course student.

In order to address this issue and still provide students with experience in interpreting results from many statistical procedures, we wrote a query language called STATAN, STatistical Analysis To Assist Neophytes. Statan is a system of CMS EXEC procedures and Fortran and Assembler programs that provides a psuedo-interactive link to major statistical packages. The system is psuedo-interactive in the sense that students respond to on-line queries which allow STATAN to implement batch versions of the statistical packages. The system was written for a IBM 370 although the principles apply to other large computers as well.

II. ADVANTAGES TO STUDENTS

The need to learn programming requirements that are package specific is a formidable task as can be seen by the size of the reference manuals and the popularity of mini-courses designed to introduce students to the statistical package. Thus, it is desirable to have a query language which can easily interact with a variety of statistical packages and stand-alone programs. Questions are written from a student's point of view. They are easily understood by novices without seeming tedious to more experienced users. At any point, the student may request additional information by typing the keyword HELP.

STATAN is geared to the way students think about statistics. If a student is interested in contingency tables or cross-tabulations, he may simply type the word contingency or cross-tabulation when asked for a procedure codename. The statistical packages which contain a cross-tabulation procedure will then be listed. The student will be asked to indicate a package and questions relevant to two or more-way tables will be posed. The student's interest is focused on the statistical method and not the wearisome details of how to make the program work.

A further advantage is the ease with which a student can specify a statistical package. STATAN provides a transparent data exchange between statistical systems. There is no need to convert the data set from the requirements of one system to the diverse needs of another. In a similar manner, students also have access to stand-alone programs which may be incorporated by the instructor. For example, the instructor may have a special program to do regression diagnostics or exploratory data analysis. Thus with minimal training students can effortlessly invoke prodigious computing power.

The concept of immediate feedback is supported by HELP messages which serve many different purposes They can be used to provide documentation of STATAN, a further explanation of queries, a description of terms used, a list of available options, or to prompt syntax usage. Additional keywords also give assistance. A complete list and some examples are presented below.

. HELP - provides a useful message
. NAMES - lists variable names
. LIMITS - identifies range of numerical entry
. OPTIONS - lists keywords which identify options
. QUIT - exit procedure
. STOP - exit STATAN package
. REPEAT - repeat question
. BACKUP - erase reply and backup to previous user response

HELP: DESCRIPTION OF MEAN COMPARISONS

```
ENTER THE MEAN COMPARISON TEST NAME (OR RETURN)
.help
ENTER ANY OF THE FOLLOWING OPTIONS (OR RETURN)
LSMEANS     IF YOU WANT LEAST SQUARE MEANS COMPUTED
            FOR THE MAIN EFFECTS.
DUNCAN & A  IF YOU WANT DUNCANS MULTIPLE RANGE TEST
            AND A SPECIFIES THE LEVEL OF SIGNIFICANCE
            FOR THE TEST
```

```
WALLER & K IF YOU WANT THE WALLER-DUNCAN T-TEST K
         SPECIFIES THE K-RATIO

PLEASE PROCEED
```

KEYWORD: OPTIONS

```
ENTER THE MEAN COMPARISON TEST NAME (OR RETURN)
.options
LSMEANS     WALLER     DUNCAN
CONTINUE TEST SPECIFICATIONS (OR RETURN)
```

HELP: VARIABLE NAMES

```
.sas.factor
ENTER THE VARIABLE LIST (NOT MORE THAN 250 VARIABLES)
.names
MOLEWT  THERMAL  MINERAL  AMINOACD  INHIBITR
CATEFFCY
PROCEED AS DIRECTED
.all
```

The HELP messages and keywords enhance the query language with memory aide that eliminates the need for recall. Rather than guess or search for a manual or other documentation, the student may find solace in typing a keyword and getting on with the business at hand.

STATAN's conversational nature helps stimulate and maintain student interest. Queries are posed about data entry, data modification, and statistical procedures. Each response is checked for validity. This immediate feedback eliminates the frustration of submitting incorrect jobs to the computer and the subsequent loss of momentum. Because STATAN transparently converts the student's responses into error-free package code and appropriate job control language, the student sees only the output generated by the statistical package. Thus, validity checks are perhaps the greatest advantage to the student. As each response is entered it is checked for errors in syntax, context, and typing. The errors are flagged with the appropriate messages of assistance. Examples of errors in syntax are: variable names exceeding eight characters, unclosed parenthesis, and incorrect use of an asterisk. Context errors are a little more difficult to detect. For example, in regression analysis a variable is either a response variable or an indicator variable. (The response variables are sometimes called predicted or dependent while the indicator variables are sometimes called predictor or independent variables.) STATAN will flag an indicator variable, if it has previously been identified as a response variable. The typing errors are flagged by checking the variable name or option specification for syntax or context. Suppose you name a variable YIELD. If in further queries you mistype or misspell the name as YEILD, the response will be flagged with the message "UNKNOWN VARIABLE NAME". (This will happen only if you do not have another variable named YEILD.)

III. DATA ENTRY

The number of concepts the student needs to understand before becoming a productive user is minimal. The student must be able to identify

1) The case or experimental unit under investigation.
2) The variables which were measured on each unit.
3) The structure and location of the data set matrix.

Within a data set, matrix variables correspond to columns and cases correspond to rows. For many students, this is the first time they have faced the necessity to identify variables, cases, and data values and to differentiate between experimental units, sampling units, and observational units. One of the most common errors students make when presented with a set of data is to overlook the categorical variable used to identify a group or class. Suppose we are given data on the amount of alcohol consumed before intoxication by men and women. The question under consideration is, "Do women have a lower tolerance to alcohol?"

Alcohol Consumption (liters)

Female	Male
3.8	3.3
2.4	7.1
6.9	4.9

For this set of data, there are two variables, sex and alcohol consumption. A case is an individual.

The information required by STATAN is the number of variables (columns), the data matrix, variable names of eight characters or less, and a title for the analysis. Data is entered into the computer one case at a time. The program is invoked by typing the word "STATAN". A sample session is given in Table 1. The computer responses are the lines in capitals and those asking for case 1?, 2?, etc. Student responses are in lower case letters which follow the computer generated period. Look at the line following "7?". The period followed by no input indicates that the return key was pressed where upon the data set was closed.

Table 1: Data Entry

```
.statan build
ENTER THE NUMBER OF VARIABLES.
.2
ENTER THE 2 VALUES FOR EACH CASE, ONE CASE AT A
TIME.  USE SPACES TO SEPARATE VALUES.
    1?
.1  3.8
    2?
.1  2.4
    3?
.1  6.9
    4?
.2  3.3
    5?
.2  7.1
    6?
.2  4.9
    7?
.
ENTER NAME FOR VARIABLE 1
.sex
ENTER NAME FOR VARIABLE 2
.alcohol
ENTER AN IDENTIFICATION NAME FOR THIS RUN OR DATA
FILE.  THIS NAME CAN BE UP TO 60 CHARACTERS IN
LENGTH.
.alcohol consumption (liters)
ENTER PROCEDURE CODENAME (HELP OR STOP)
```

The data may be entered in one of three ways: directly from the terminal as illustrated in Table 1, from storage disks, or randomly generated from 20 common distributions.

butions. The data on the storage disk may be a file created from floppy disk, punched cards, or a data file previously saved by STATAN. This option allows the instructor to enter a large data matrix to be used by the entire class. The data matrix is saved under a filename and each student invokes that set of data by typing the filename.

Table 2: Invoking Previously Stored Data

```
ENTER NAME OF PREVIOUSLY SAVED DATA FILE OR
     'BUILD' TO CREATE A NEW FILE OR
     'SAS.GENERATE' TO GENERATE DATA FROM PROBABIL-
     ITY DISTRIBUTIONS OR
     'PROCEDURES' TO SEE WHAT STATISTICAL PROCE-
     DURES ARE AVAILABLE.
.census
FILE CENSUS STATAN COPIED TO WORK SPACE.
ENTER PROCEDURE CODENAME (HELP OR STOP)
```

The generate option helps students get a feeling for sampling variation. For instance, a student may generate 100 samples of size 5 from a normal distribution with a specified mean and standard deviation. Sample means and confidence intervals may be calculated and plotted. The student can then see how many times the true parameter is covered. Another concept that can be easily illustrated is the central limit theorem. The student may generate random samples from non-normal distribution and plot the means or do a goodness-of-fit test.

IV. AVAILABLE PROCEDURES

The list of procedures currently available in STATAN may be obtained by entering the word "Procedures". At present there are four categories of procedures, Data Management, Data Modification, and the statistical procedures SAS and SPSS.

```
statan
ENTER NAME OF PREVIOUSLY SAVED DATA FILE OR
     'BUILD' TO CREATE A NEW FILE OR
     'SAS.GENERATE' TO GENERATE DATA FROM PROBABIL-
      ITY DISTRIBUTIONS OR
     'PROCEDURES' TO SEE WHAT STATISTICAL PROCEDURES
     ARE AVAILABLE.
.procedures
WHICH SECTION WOULD YOU LIKE TO SEE? (HELP, ALL,
OR NONE?)
.all
CODENAME    SPSS PROCEDURE
  BREA      BREAKDOWN
  CCOR      CANONICAL CORRELATION
  COND      CONDESCRIPTIVE
  CROS      CROSSTABS
  DISC      DISCRIMINANT ANALYSIS
  FACT      FACTOR ANALYSIS
  PC        PRINCIPAL COMPONENTS ANALYSIS
  FREQ      FREQUENCY
  NONP      SPEARMAN & KENDALLS NONPARAMETRIC
            CORRELATION COEFFICIENT
  ONEW      ONEWAY ANALYSIS OF VARIANCE
  PART      PARTIAL CORRELATION COEFFICIENT
  PEAR      PEARSON CORRELATION
  REGR      REGRESSION
  SCAT      SCATTERGRAM
  T-TE      T TEST
  SPSS      ENTER ANY VALID SPSS PROCEDURE STATEMENT
  SAS       ENTER ANY VALID SAS PROCEDURE STATEMENT
  SAS.GETP  ENTER FILE NAME FOR PREVIOUSLY BUILT
            SAS PROCEDURE STATEMENTS
  CRD       COMPLETELY RANDOMIZED DESIGN ANOVA
RCB         RANDOMIZED COMPLETE BLOCK DESIGN ANOVA
FEXPT       FACTORIAL EXPERIMENT ANOVA
LATI        LATIN SQUARE DESIGN ANOVA
NEST        NESTED ANALYSIS OF VARIANCE
SPLIT PLOT  SPLIT PLOT DESIGN
SAS.FREG    FREQUENCY
SAS.CROS    CROSSTABS
SAS.PC      PRINCIPAL COMPONENTS
SAS.FACT    FACTOR ANALYSIS
SAS.CANO    CANONICAL CORRELATION
SAS.DISC    DISCRIMINANT ANALYSIS
SAS.NEIG    NEAREST NEIGHBOR DISCRIMINANT ANALYSIS
SAS.CLUS    HIERARCHICAL CLUSTER ANALYSIS
SAS.TTEST   T-TEST ANALYSIS
SAS.CORR    CORRELATIONS ANALYSIS
SAS.REGR    REGRESSION
SAS.NPAR    ONE-WAY ANALYSIS OF VARIANCE ON RANKS
SAS.MEAN    SIMPLE UNIVARIATE DESCRIPTIVE STATISTIC
            FOR NUMERIC VARIABLES
SAS.RANK    COMPUTE RANKS FOR NUMERIC VARIABLES,
            CREATING A NEW DATA SET CONTAINING
            THE RANKS IN PLACE OF THE ORIGINAL
            VALUES
SAS.PLOT    GRAPHS ONE VARIABLE AGAINST ANOTHER,
            PRODUCING A PRINTER PLOT
SAS.SPLOT   SIDE-BY-SIDE SCHEMATIC PRINTER PLOTS
SAS.PRINT   LISTS THE VALUES OF SOME OR ALL THE
            VARIABLES IN A DATA SET
SAS.SORT    REARRANGE CASES IN A SAS DATA SET OR
            CREATE A NEW DATA SET CONTAINING THE
            REARRANGED CASES
SAS.CHART   PRODUCES VERTICAL AND HORIZONTAL BAR
            CHARTS, PIES, BLOCK, AND STAR CHARTS
SAS.FUNCAT  MODEL FUNCTIONS OF CATEGORICAL
            RESPONSES AS A LINEAR MODEL
SAS.UNIV    SIMPLE DESCRIPTIVE STATISTICS FOR
            NUMERIC VARIABLES
SAS.GLM     GENERAL LINEAR MODEL

            DATA MANAGEMENT PROCEDURE

DROP        DROP A CASE OR VARIABLE FROM YOUR
            DATA FILE
ADD         ADD A NEW VARIABLE OR ADDITIONAL CASES
REPLACE     REPLACE AN OBSERVATION OR CASE
LIST        LIST A SPECIFIC CASE, VARIABLE, OR
            OBSERVATION
CHANGE      CHANGE THE NAME OF A VARIABLE
SAVE        SAVES DATA FILE AND ASSOCIATED
            INFORMATION
       NOTE:DATA IS RETAINED ON CLASS STORAGE FOR
            ONE MONTH FROM DATE OF LAST SAVE
            OPERATION
DELE        DELETES DATA FILE FROM SYSTEM STORAGE
REST        RESTART THE STATAN SYSTEM

            DATA MODIFICATION PROCEDURE

RECODE      RECODE OBSERVATIONS TO NEW VALUES
MISSING     TREAT SOME OBSERVATIONS AS MISSING
            VALUES
COMPUTE     COMPUTE STATEMENTS
SELECT      SELECT SUBSET OF DATA FOR ANALYSIS
MODCOR      MODIFY CORRELATION MATRIX
IF          BUILD IF-THEN STATEMENTS
SUM         SUM A LIST OF VARIABLES
COUNT       COUNT VALUE OCCURRENCES IN OBSERVATIONS
*SAS        ANY VALID SAS PROGRAM STATEMENT
*SPSS       ANY VALID SPSS PROGRAM STATEMENT
TITLE       CHANGE THE TITLE ON THE DATA SET
CHANGE      FOR CHANGING VARIABLE NAMES
MODDELETE   DELETE A MODIFICATION
```

```
MODLIST    TO LIST ALL MODIFICATIONS THAT ARE
           CURRENTLY CONTAINED IN A STATAN
           DATASET
*****************************************************
***NOTE:  ALL THE ABOVE MODIFICATION PROCEDURES****
***MAY BE ABREVIATED TO FOUR CHARACTERS OR MORE.***
*****************************************************
```

Under the heading DATA MANAGEMENT are actions performed on the data file as a whole rather than modifications within a given data set. After all statistical analyses are completed on one set of data it is often desirable to switch to another data set. STATAN enables the user to save data sets, concatenate two or more data sets, and delete data sets from class storage. Password protection is provided for those who want to be sure unauthorized users do not write over their data set.

Data Modification differs from data management in that modifications are made within a given data set. It is frequently desirable to edit the data matrix by adding (dropping or replacing) a case (variable or observation), to create a new variable from an arithmetic or logical expression of one or more existing variables, or to recode a variable to new values. The student decides what to do; then STATAN prompts how to do it.

V. STATISTICAL PROCEDURES

At present there are two types of statistical procedures, SAS and SPSS. These procedures are interactive versions of the data analysis routines provided by SAS Institute or SPSS Incorporated. The student specifies the routine and the computer asks for additional information such as a list of variable names. (Table 3)

Table 3: SPSS Procedure T-TEST

```
ENTER PROCEDURE CODENAME (HELP OR STOP)
.spss.ttest
DO YOU WANT A T-TEST WITH INDEPENDENT SAMPLES
OR A PAIRED T-TEST (I or P?)
.help
INDEPENDENT SAMPLES OCCUR WHENEVER INDIVIDUAL
UNITS ARE DRAWN FROM ONE SAMPLE WITHOUT REGARD
TO THOSE DRAWN FROM THE OTHER SAMPLE.

PAIRED SAMPLES OCCUR WHENEVER AN INDIVIDUAL
FROM ONE SAMPLE IS SELECTED ALONG WITH A
SIMILAR INDIVIDUAL FROM THE OTHER SAMPLE.

DO YOU WANT A T-TEST WITH INDEPENDENT SAMPLES
OR A PAIRED T-TEST.
.i
ENTER THE CLASS VARIABLE (ONLY ONE).
.help
CLASS VARIABLES ARE ALSO KNOWN AS FACTORS,
DISCRETE VARIABLES, NOMINAL VARIABLES, CATEGOR-
ICAL VARIABLES, AND QUALITATIVE VARIABLES; THEY
IDENTIFY THE GROUPS FOR THE ANALYSIS.

ENTER THE CLASS VARIABLE (ONLY ONE).
.sex
ENTER TWO VALUES THAT IDENTIFY THE GROUPS.
.1  2
ENTER THE RESPONSE VARIABLES.
.alcohol
CONTINUE VARIABLE LIST (OR RETURN)
.
PROGRAM ENGAGED........PLEASE WAIT
```

VI. CONCLUSIONS

STATAN is available for use on an IBM 370 running under VM/CMS. Readers interested in obtaining this system should contact the author.

STATAN has been used with great success by several instructors of statistics courses at University-Lincoln. Evaluation from both participating teachers and students helped to determine the merits of STATAN as well as pinpoint areas in need of improvement. On open-ended response questions, the system was enthusiastically endorsed. The students appreciated the opportunity to analyze many sets of data and thus gain invaluable experience in interpreting statistical results. By eliminating the need for lengthly calculations students can focus on such important points of the analysis as the hypothesis being tested, the meaning of the probability statements, and the research conclusions. Students can investigate what happens when the assumptions underlying the statistical analysis are violated. They begin to realize why it is important to address these questions when planning experiments.

Areas pinpointed for improvement were the terminology and phraseology used in the queries and "help" messages. This prompted us to revise the system so that queries and "help" messages are now stored as two data files separate from the rest of the system. Each instructor may then use the terminology he or she deems most appropriate. However, all agree that easy access to the powerful statistical packages helps the student explore concepts and gain a deeper understanding of statistics.

AN EMPIRICAL ANALYSIS OF "LEARNIT," A MICROCOMPUTER PROGRAM FOR FACILITATING MEMORIZATION[1,2]

Gary L. Schilmoeller
Kathryn J. Schilmoeller
Dennis R. King
University of Maine
Orono, Maine 04469
207 581 7171

Ideally, the creation of software for educational purposes should involve a multidisciplinary effort. Professional programmers, trained in computer science, have the task of writing the software to teach some subject (Chinese, for example) and evaluating its effectiveness in terms of compatibility with the computer hardware. These professional programmers may or may not have expertise in education or in the particular discipline for which they are programming. The software users, however, often will be from other disciplines (in this example, foreign languages). Evaluation of software from their perspectives will focus on a different set of criteria. For the software user, the software must be functional not only in terms of controlling the computer, but also in terms of facilitating student learning. If the professional programmer and the educator can collaborate during the actual creation of the software, both sets of criteria can be combined to rigorously evaluate the software before it is disseminated for public use.

Unfortunately, many educators do not have the financial resources or professional contacts to commission professional programmers to write programs tailored to their particular discipline. Instead, these educators must either write their own programs or must survey existing software available commercially or in publications, and must make decisions regarding which existing programs might fit their specific needs. Failure to determine the functionality of software in terms of its effect on student learning may have undesirable side effects. An educator who tries a program on the assumption that it successfully teaches, and finds that students do not learn, may become a recalcitrant computer user in the future or, worse, may lobby actively against the use of computers in the educational process.

In a recent article on the applicability of microcomputer technology in special education, Florence Tabor (1981) offered an evaluation format to assist potential software users in their search for functional programs. The evaluation format is divided into two main categories--instructional content and organization/implementation of program materials. Specific questions in each of these categories guide the educator in evaluating the software in relationship to current theories of learning and motivation. (The complete evaluation form is printed in the Tabor article.) This questionnaire represents a valuable contribution in terms of conceptual evaluation. Careful screening of software using this questionnaire should help educators make better program decisions, thus reducing the probability of selecting nonfunctional programs. However, the ultimate test of functionality from the educational perspective needs to be an empirical test. This paper, whose authors represent the disciplines of child development, early childhood learning, and special education respectively, addresses the issue of software evaluation at both the conceptual and empirical levels.

Recently, the second author of this report came across a published program called "Learnit" (Carlson, 1980) that was offered as a useful memorization aid. This program seemed particularly appealing to educators who often want to impart a basic core of facts, definitions, formulae, vocabulary lists, or theories that will provide a basis for subsequent expertise and fluency in a particular discipline.

The "Learnit" program combines stimulus control procedures from an operant psychology paradigm with microcomputer technology. A teacher or student enters the material to be memorized into the computer. As presented in the article, when a student uses the "Learnit" program, the initial instructions read:

> Press RETURN to see passage. When it appears, read it aloud very carefully. Then press RETURN again, and some of the letters will disappear. Read the passage again, and then press RETURN to eliminate some more letters. In ten passes all the letters will be gone. By then you will have it all memorized. (Carlson, 1980, p. 74)

This random delete procedure is similar to the superimposition and fading procedures used by Corey and Shamow (1972) to teach young children oral reading. For example, they initially superimposed a picture of a cat over the printed word "cat." Over a series of trials, the picture was faded out gradually until finally the child was seeing only the printed word "cat." In both this fading procedure and in "Learnit," some visual cue is introduced initially to help students make a response they otherwise could not make. Gradually, the added cues are removed, so that the student has to respond to the criterion stimulus--the word "cat" for Corey and Shamow, or the instruction to recall a designated passage when "Learnit" is used.

A number of researchers from the learning theory tradition have noted the difficulty of using fading procedures successfully (e.g., Bijou, 1968; Gollin & Savoy, 1968; Schilmoeller & Etzel, 1977; Schilmoeller, Schilmoeller, Etzel & LeBlanc, 1979; Sidman & Stoddard, 1966). Some students attend to the cue that is being faded rather than to the stimulus that ultimately should evoke the correct response. For example, in the Corey and Shamow example, some children would attend to the picture of the cat until it was faded completely. At that point they would not be able to read "cat" because they had never attended to the printed word. Evaluating "Learnit" in the light of this body of research, a similar result seemed plausible. Students using "Learnit" might be able to correctly recall the target text as long as any letters remained on the CRT monitor to cue their memory. When all the letters disappeared, they might go as blank as the screen they faced.

With this possibility in mind, and in the absence of any available field data attesting to the actual success of "Learnit," an evaluation study seemed in order. Three questions were asked. First, would students successfully memorize a passage using the "Learnit" program? Second, would the "Learnit" facilitate memorization more successfully than a more traditional trial-and-error procedure that also utilized the computer? Third, in the event that a range of scores was obtained regardless of which procedure was used, could some predictors of success be identified, such a grade point average, previous experience with computers or other mechanical devices, or self-evaluation regarding ability to memorize?

METHODS

Subjects. Forty undergraduates from an introductory course in child development at the University of Maine at Orono served as subjects. These were predominately sophomores and juniors (mean years of completed education = 14.6) with a mean age of 20.6 years (range 18 to 35). By random assignment, half of these students experienced the "Learnit" program and half experienced a control procedure.

Apparatus and Programs. A TRS-80 Model I microcomputer and Model VI printer were used. The text to be memorized was selected from John Steinbeck's Grapes of Wrath (1939). The content of the passage dealt with the socialization influences of families, a topic of discussion in the Child Development course.

"Learnit." For this study, "Learnit" began with the following instructions on the CRT:

> You will be memorizing a paragraph by John Steinbeck. To help you with this task, the paragraph will be displayed on this screen ten times. Each time the text appears, read it aloud. When you have finished and are ready to go on to the next display, push ENTER. There will be a 90 second delay between each display.
> Push ENTER when you are ready to begin.

These instructions were modified from the original instructions for two reasons. First, the stimulus control literature focused on the manipulation of the visual stimuli as a potentially helpful or disruptive variable and by changing the instructions in this study, the experimental question could focus on this issue directly. Second, the same set of instructions could be used for both the experimental and control procedures in this study, thus controlling an additional source of variance in the results.

Subsequent to reading the initial instructions and pressing ENTER, the first trial of the program began with the complete text printed on the CRT. After reading the text aloud at least once, the students pressed ENTER which cleared the screen. At that time the computer randomly removed letters from the text during the 90-sec intertrial interval that was controlled by turning off the CRT. At the end of this interval, the CRT was turned on and the text appeared minus the randomly deleted letters. Table One compares the text as it appeared on the first and last trials of the "Learnit" program. Ten

TABLE 1

First and last trials from the "Learnit" program.

```
-----------------------------------------

   IN THE EVENING A STRANGE THING HAPPENED:
THE TWENTY FAMILIES BECAME ONE FAMILY: THE
CHILDREN WERE THE CHILDREN OF ALL.  THE
LOSS OF HOME BECAME ONE LOSS, AND THE
GOLDEN TIME IN THE WEST WAS ONE DREAM.
AND IT MIGHT BE THAT A SICK CHILD THREW
DESPAIR INTO THE HEARTS OF TWENTY FAMILIES,
OF A HUNDRED PEOPLE: THAT A BIRTH THERE IN
A TENT KEPT A HUNDRED PEOPLE QUIET AND
AWESTRUCK THROUGH THE NIGHT AND FILLED A
HUNDRED PEOPLE WITH THE BIRTH-JOY IN THE
MORNING.  A FAMILY WHICH THE NIGHT BEFORE
HAD BEEN LOST AND FEARFUL MIGHT SEARCH
ITS GOODS TO FIND A PRESENT FOR A NEW
BABY.  IN THE EVENING, SITTING ABOUT THE
FIRE THE TWENTY WERE ONE.  THEY GREW TO
BE UNITS OF THE CAMPS, UNITS OF THE EVEN-
INGS AND THE NIGHTS.  A GUITAR UNWRAPPED
FROM A BLANKET AND TUNED-AND THE SONGS,
WHICH WERE ALL OF THE PEOPLE, WERE SUNG
IN THE NIGHTS.  MEN SANG THE WORDS, AND
WOMEN HUMMED THE TUNES.

-----------------------------------------

      T       I         G        H     E
           F             E O
              TH            F
L S                  N
   D         N        T     O
          H  B              C
         N        EA
     H N
                I             R    N
M                  W
      E    S           U         E  C
      O
   Y   I         N N ,          BO
                            H
 E        F               TS
I                               N     E
   M    L               -
  I         L              E
     E      S     N              S
       UM
```

Source: John Steinbeck. The Grapes of Wrath. New York: The Viking Press, 1939, pp. 264-265.

trials were preprogrammed to accomplish the total deletion of the passage. Each student in the "Learnit" group went through all ten trials or worked on the task for one hour, whichever occurred first.

Trail-and Error Control. Students in the control group experienced the same interaction with the computer, with one exception. Each time the next appeared after an intertrial interval, the complete text was displayed. This would be considered a more traditional trial-and-error method of learning. Again, each student went through ten trials or worked for one hour, whichever occurred first.

Procedure. When students indicated agreement to participate in this study by signing an informed consent form, they also filled out a questionnaire that provided information on variables thought to be potential predictors of successful memorization. These variables included age, educational level, GPA, self-ratings regarding ability as students and as memorizers, frequency of experience with mechanical devices often used as education aids (cameras, calculators, video equipment, computers, etc.), and perceived competency with such mechanical devices.

After completing this questionnaire, students were assigned randomly to one of the two programs and scheduled for their sessions with the computer. When they finished their interaction with the computer program, they were asked to recite and to write out what they recalled of the text. From the written recall passages, comparisons between groups were made on: 1) total number of words written; 2) total number of correct words (defined as number of text words including legitimately repeated text words); 3) total number of text words (including text words that were repeated more times than actually occurred in the text); and 4) number of correct text clauses.

RESULTS

As shown in Table Two, t-tests of the differences between means on the measures of written recall resulted in either no significant differences between groups or significantly better performance by the control group students. Students who experienced the full-repeat control procedure wrote more words, $t(1,38)= -2.14$, $p < 0.039$, and had more correct text words, $t(1,38)= -2.06$, $p < 0.047$, than did students who experienced the random delete "Learnit" procedure. Comparisons of total text words including surplus (and thus incorrect) repeats and of total correct clauses resulted in no significant differences between groups, though in each instance the means of the full-repeat control group were larger than the means of students from the "Learnit" group.

Measured against absolute standards of recall, neither group's performance was impressive. Students who experienced "Learnit" had a mean of 88.8 correct test words out of a possible 166 (53.3%), as seen in Table Three. Students who had the full repeat control program averaged 114.5 correct test words (68.9%). Of the thirteen possible clauses, "Learnit" students wrote an average of 1.95 (15%) correctly and control students wrote 3.2 (24.6%) correctly.

Since some individual students in each group were very successful memorizers, correlational analyses were done to determine whether any of the questionnaire

TABLE 2

Group means and standard deviations on written recall test.

	Group						
Recall Measures	"Learnit"		Control				
	$\bar{x}$	(SD)	$\bar{x}$	(SD)	t-value	df	
Total Words	100.25	(50.12)	128.75	(31.98)	-2.14	1,38	$p < 0.039$
Correct Text Words	88.85	(44.95)	114.50	(33.07)	-2.06	1,38	$p < 0.047$
Text Words Plus Surplus Repeats	95.00	(49.16)	120.15	(34.18)	-1.88	1,38	$p < 0.068$
Correct Clauses	1.95	(2.72)	-3.20	(2.23)	-1.59	1,38	$p < 0.121$

items were positively associated with successful memorization. No significant correlations existed, thus, precluding at this time an enumeration of potential predictor variables that could be used to select candidates for interactions with the computer in this memorization facilitation exercise.

TABLE 3

Mean Percentage Correct on Written Recall Test

Recall Measures	"Learnit" (N = 20)	Control (N = 20)
Correct Text Words	53.5	68.9
Correct Clauses	15.0	24.6

DISCUSSION

These data present negative answers to each of the three questions asked in the study. "Learnit," as used in this study, did not result in successful memorization. Further, compared to the full-repeat control procedure, students using "Learnit" did no better, and, by some measures, significantly worse, as memorizers. Finally, none of the descriptive variables correlated with students' ability to memorize.

Gollin and Savoy (1968), Koegel and Rincover (1976), and other researchers had noted that progressive manipulations of the visual stimuli (as in fading or in the random delete procedure) do not always facilitate learning. That was the case in this study, too. Yet there is an important difference in where the errors occurred in the stimulus control studies and in this study. In studies using fading, errors usually occurred on the final trials of the program, when the fading cue was nearly gone as a discriminative cue for the correct answer. By contrast, in this study, students experiencing "Learnit" had difficulty almost as soon as the random deletion began. This was particularly perplexing since the authors had experienced success when piloting "Learnit" before the actual study began. One possible explanation for the disruption experienced by the "Learnit" subjects might be in the modification of the instructions for this study. Pilot subjects all knew that letters would be deleted at random, as the experimental subjects would have known had the original instructions been utilized. From the results of this study, one clear conclusion is that the stimulus manipulation (i.e., the random deletion of letters) by itself was insufficient to facilitate memorization. Further evaluation must be made to determine whether the random deletion in conjunction with the original instructions will be more successful.

Two additional program revisions were suggested by error analyses. First, a back-up capability could be included in the program. As published, if one made mistakes, "Learnit" had no mechanism allowing one access to the complete passage to correct errors. Carlson (1980) suggested such a procedure as a possible addition to his program, although it was not included in the published version. A back-up procedure might enable students to review the correct words and then progress successfully through the program. Second, asterisks or some other special character might be substituted as place-markers for the deleted letters. By using this step, students would be given the opportunity to connect the remaining letters on any given step into a visual configuration that might help cue the total word.

Given the results of this study, have the authors become recalcitrant or opposed to the idea of using computers in education? Not at all! Etzel, LeBlanc, Schilmoeller, and Stella (1980) have argued convincingly that developers of educational technologies must bear the burdens of student errors. They note that students who have difficulty in conceptual development often learn when such variables as curriculum materials, environmental arrangements, and instructions are modified. Student performance evaluates the success or failure of such modifications. In this study, the students' responses indicated that this version of "Learnit" was not functional. Now the task is to determine how to modify this software so that it will not only control the computer hardware, but functionally assist the students' learning as well. When both the conceptual and empirical tests of a good program are met, the technologies of both computer specialists and educators will be satisfied, and dissemination of the product can proceed from a sound empirical base.

FOOTNOTES

[1] This research was partially supported by funds from a Faculty Research Grant from the University of Maine, from the University of Maine College of Education, and from the Maine Life Sciences and Agriculture Experiment Station.

[2] The authors gratefully acknowledge the assistance of Judy Feeney and Jay McIntire during the data collection phase of this project.

REFERENCES

Bijou, S. W. Studies in the experimental development of left-right concepts in retarded children using fading techniques. In Norman R. Ellis (Ed.) *International Review of Research in Mental Retardation*. (Vol. 3). New York: Academic Press, 1968.

Carlson, Neil R. Learnit: A memorization-aid program. *onComputing*, *1981*. *3*, 72-78.

Corey, J. R. & Shamow, J. C. The effects of fading on the acquisition and retention of oral reading. *Journal of Applied Behavior Analysis*, 1972, *5*, 311-316.

Etzel, B. C., LeBlanc, J. M., Schilmoeller, K. J. & Stella, M. E. Stimulus control procedures in the education of young children. In S. W. Bijou and P. Ruiz (Eds.), *Contributions of Behavior Modification to Education*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1980.

Gollin, E. S. & Savoy, P. Fading procedures and conditional discrimination in children. Journal of the Experimental Analysis of Behavior, 1968, *11*, 443-451.

Koegel, R. L. & Rincover, A. Some detrimental effects of using extra stimuli to guide learning in normal and autistic children. *Journal of Abnormal Child Psychology*, 1976, 4, 59-71.

Schilmoeller, K. J. & Etzel, B. C. An experimental analysis of criterion- and noncriterion-related cues in "errorless" stimulus control procedures. In B. C. Etzel, J. M. LeBlanc, & D. M. Baer (Eds.) *New Developments in Behavioral Research: Theory, method, and application*. *In Honor of Sidney W. Bijou*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1977.

Schilmoeller, G. L., Schilmoeller, K. J., Etzel, B. C. & LeBlanc, J. M. Conditional discrimination after errorless and trial-and-error training. *Journal of the Experimental Analysis of Behavior*, 1979, *31*, 405-420.

Sidman, M. & Stoddard, L. T. Programming perception and learning for retarded children. In N. R. Ellis (Ed.) *International Review of Research in Mental Retardation*. (Vol. 2). New York: Academic Press, 1966.

Steinbeck, John. *The Grapes of Wrath*. New York: The Viking Press, 1939.

Tabor, Florence. The microcomputer-Its applicability to special education. *Focus on Exceptional Children*. 1981, *14*, 1-14.

Special Session

EDUCATION FOR CAREERS IN HEALTH COMPUTING

Sponsored by: HEN & SCM
Chaired by: Karen A. Duncan
Health Information Systems

ABSTRACT

This session explores in great depth the career opportunities in health computing that would require Associate, Bachelors, or M.S. level education. The panelists will discuss educational requirements and how these requirements might be met by people entering in from another field such as education, nursing, or computer science. Actual & proposed formal curricula will also be presented.

Participants

Richard Austing
University of Maryland

Richard E. Pogue
Medical College of Georgia

Special Session

COMPUTERS IN PUBLIC SCHOOL CLASSROOMS: TEACHER TRAINING ISSUES AND CONCERNS

Sponsored by: SIGCSE
Chaired by: Sandy Pratcher
Texas Educational Agency

ABSTRACT

Training and experience requirements for teaching computer-related courses are generally lacking. Formal teacher training programs are few and far between. Marvin Minski of the Massachusettes Institute of Technology has said: "Eventually, programming itself will become even more important than mathematics in early education." If he is correct, educators may have serious problems ahead unless teacher training in computer literacy is incorporated into both the pre-service and in-service requirements of teachers.

In a survey conducted by the Minnesota Educational Computing Consortium, 85% of 1300 teachers surveyed agreed or strongly agreed that secondary school students should have minimal understanding of computers. However, only 39% of that group agreed or strongly agreed that their own training was adequate for using computers in instruction.

In Illinois, J. Richard Dennis surveyed 686 secondary school principals and found that 71% saw a need for computer science teachers, 55% saw a need for state certification in computer science, and 82% felt that some background in computer science is valuable for any teacher.

Several national curriculum organizations have called for training teachers to use computers in schools. For example, the National Council of Teachers of Mathematics, in its recommendations for school mathematics of the 1980's recommends that "All mathematics teachers should acquire computer literacy either in preservice programs or through inservice programs funded by school districts in order to deal with the impact of computers on their own lives and to keep pace with the inevitable sophistication their students will achieve."

Stuart Milner, in his article for Phi Delta Kappa (April, 1980) entitled "Teaching Teachers about Computers: A Necessity for Education" summarizes the need for teacher training: "...teachers and administrators must recognize how essential inservice training is for effective use of computers in instruction. While the amount of training may vary, some sophistication is necessary. Given the proliferation of microcomputers, unless serious attention is given to upgrading teachers' competencies, students may well become more computer literate than their teachers.

Participants

Karen Billings
Columbia University

George Culp
Univ. of Texas at Austin

Arthur Luehrmann
Computer Literacy

Jim Poirot
North Texas State University

Project Presentation

COMPUTER AWARENESS PROJECTS

Chaired by: Dennis Spuck
University of Houston

Petra Lucia Arrieche
Matilda Butler
Douglas Carr
Rowena Hacker
Beverly Hunter
Cynthia A. Lunning
Marice Mitchell
William Schnipert
Susan M. Zgliczynski

ABSTRACT

Problem Instruction for Microcomputers

Petra Lucia Arrieche, FUNDESED, Calle Zury, Patarapa Barquisineto, Venezuela

A project to introduce microcomputers into the author's school is described. The program is directed toward non-computer users. One of the unanswered questions at this time is the appropriate level of instruction.

ABSTRACT

Electronic Information Officer

Matilda Butler, Chair, Technology And Communication Dept., Far West Laboratory for Educational Research and Development, 1855 Folsom St., San Francisco, CA 94103

Issues concerning the computer-user interface surfaced during a project designed to develop and implement an "electronic information officer" that could convey routine information to visitors to an educational research and development organization. With microcomputers becoming widely available in elementary and secondary education institutions, this is an application that can free staff from the burden of relating routine and repetitive information and can even demonstrate the school's use of computers for teaching and administration.

During the development phase, we learned that the successful use of Electronic Information Officer depended on software that is appropriate for the first-time or occasional user and requires hardware that is "crash proof." From the institution's point of view, the software also needs to provide for easy updating of information and must be accessible by visitors without staff assistance.

There was an Atari 400 and a color monitor available for this use. The software was developed to take advantage of color, sound, and graphics. A menu of choices also helped to facilitate use. When the Electronic Information Officer was installed, the major problem turned out to be the hardware rather than the woftware. Visitors could make the system "crash" by hitting the ESC, Break, or Reset keys. The Atari 400's membrane keyboard made it difficult to disable individual keys.

The solution turned out to take advantage of one of the ports in the front of the Atari that is normally used for a joystick or paddles. We developed a nine-button box with an Atari compatible plug that enabled the visitor to use the program without using the keyboard. The actual keyboard was put inside the pedestal that had been built to support the monitor. The user only had access to the nine-button box. A red button could be pressed to interrupt the continuing graphic and music display and would bring up the menu. Eight black buttons that corresponded to the eight items on the menu could be pressed for more information about each item.

This project raised some larger issues surrounding the computer-user interfact. What kinds of software and hardware are needed to ease access to computers in the absence of computer literacy? One framework being developed uses The Johari Window (Joseph Luft, Of Human Interaction, Palo Alto, CA: National Press Books, 1969). Initially used to examine interpersonal relationships, the four cells -- Open (known to self and known to computer), Blind (not known to self and known to computer), Hidden (known to self and not known to computer), and Unknown (not known to self and not known to computer) suggest some factors that may hinder or facilitate use of the computer.

ABSTRACT

A Computer Elective Course for Seventh and Eighth Graders

Rowena Hacker, Los Alisos Intermediate School, 25171 Moor Ave., Mission Viejo, CA 92691

The computer elective course was made possible in 1978 through a grant of $6500 from Title IV-B funds administered through the Saddleback Valley Unified School District, of which Los Alisos Intermediate Ochool is a part. Approximately $1500 was funded through the school budget. The hardware purchased includes three Horizon systems with disc drives, three MicroAge keyboards, three 9-inch Sanyo monitors, one 17-inch Hitachi Minitor and a DECwriter.

The following year an additional $2500 was funded through Title IV-B and approximately $1000 through the school budget. This allowed the purchase of four 8K Commodore PET's. On a special "buy two, get one free" promotion, we received a total of six PET's (four purchased, two free) to bring the school's microprocessor inventory to nine.

The final year of Title IV-B funding brought $3500 to purchase two 8K and two 16K PET's. The educational offer was again in effect to give us an additional six computers for th price of four. The school budget allocated $900 for the purchase of a PET printer.

The computer elective class was designed to introduce students of all ability levels to computer literacy. Students are introduced to the history of computing devices, parts of a computer, uses of the computer in everyday life, flowcharting, careers in the data processing industry, the binary number system, and beginning programming in BASIC. The classroom activities include working from one of the three different class sets of textbooks, viewing a sound filmstrip, participating in a discussion, working with a template to produce an orderly flowchart, or working on-line.

Students rotate through the various activities to complete a unit. The class begins a unit together with some group instruction for one day, then they work through the assignments for that unit. All assignments are due at the end of the specified length of time given to complete the "contract". The following day is spent reviewing the work that was done to prepare for the unit test. Eight teacher-made units are completed by the end of the twelve-week course.

To encourage many students to become acquainted with the computers, the computer lab is open before and after school for student use. Students enrolled in the computer class may use the system during these hours for work necessary to complete a unit, but the systems are used primarily by students who are not currently enrolled in a math elective, but wish to use the computers. The added "free" time on the computers can foster much learning and creativity for the students who are actually working on-line and also for their friends, who inevitably come to watch them.

The students are not the only people at the school who are learning more about computers. Teachers from many different departments have learned to use the systems for recording grades, printing letters about student progress to send home to parents, printing up lists of scores and grades to be posted for students, simulation games, and computer-assisted instruction. It is remarkable to see the motivation of some students increase because their English teacher or P.E. coach has been seen using the computers. Students not only take more interest in the computers, but are more motivated in that subject area as well.

With the support of the administration, faculty, students, and parents, the computer course has grown in equipment and enrollment. The benefits of completing the course have been and will continue to be many, from advanced placement in the high school courses in computer programming to career interest to understanding how computers affect our daily lives.

ABSTRACT

SDIP Activity III

Dr. Maurice Mitchell, Jr., Douglas Carr, Cynthia A. Lunning; Des Moines Area Community College, 2006 S. Ankeny Blvd., Ankeny, Iowa 50021

Des Moines Area Community (DMACC) was awarded a Title III Higher Education grant in June, 1981. Activity III - Instructional Development is one segment of the larger grant.

The mission ofr this project is to develop eight individualized instructional modules assembled as a correspondence course and targeted toward non-traditional (non-congregate) students to facilitate their educational success and articulation into mainstream educational participation. The instruction will be credit-bearing and creditable toward DMACC degree requirements. The college deans of the academic divisions selected the curriculum areas of personal computers and computer literacy.

Activity III tasks encompass the preparation of objectives, identification of instructional content, preparation and writing of supportive materials, preparation and evaluative instruments, and formative evaluation representing the faculty, began September 1, 1981, with the specifications of the strategy for the needs analysis survey and the data reduction needed to identify the target population and the terminal objectives.

Terminal objectives for the course were determined to be: to develop computer literacy and to assist in the selection and use of a personal computer system based upon the individual user's needs. The learning analysis, hierarchy of skills and enabling performance objectives were drafted after review of the needs analysis and study of appropriate materials from the project's literature search.

The contents of the instructional modules include the types of computers, computer systems, hardware and software, computer functions, capabilities and limitations of computers, practical applications for the home and for business, computer-aided decision making, computer language and programming, and selection and expansion of a personal computer system to best meet the user's needs. The literature search and resultant reference list with cross-references will be completed during the final stages of module development. The reference list will be included with the course materials.

Computer terms are explained in context in both the textbook and the supportive materials. Instructional activities include assigned readings from course materials, recommended reading from the reference list, written activities, and self-checks at the ends of the modules. Evaluation of the individual's progress includes a mid-course examination, a writing assignment, and a final examination over the entire course. Self-evaluation will occur in the form of self-checks.

Introductory in content, this correspondence course will be integrated into the college curriculum at the freshman level in June, 1982, with the expectation of extending the college services to those individuals not presently being served by an educational institution.

ABSTRACT

Computer Literacy in a Small Liberal Arts College

William Schnippert, Assistant Professor,
Elmira College, Elmira, NY 14901

- I. FACULTY: THE KEY TO COMPUTER LITERACY
  - A. Failure of Introductory Workshop
  - B. Humanities - vs - Professions
  - C. Literacy Luncheons
  - D. Second Introductory Workshop
  - E. Faculty Children Involvement
  - F. Release Time Problem
  - G. Microcomputer "loaners"
  - H. Software Search Problem
- II. STUDENT COMPUTER LITERACY: MANDATED OR PURSUED
  - A. State-mandated Requirements
  - B. Undergraduate Majors involving Computers
  - C. Undergraduate Majors not involving Computers
  - D. Graduate majors and computer Problems
  - E. Computer Club
- III. PERSONNEL: THE DIFFERENCE BETWEEN FUN AND FRUSTRATION
  - A. Faculty
  - B. Computer Center Staff
  - C. Librarians
- IV. FACILITIES: A STUDY
  - A. Computer Image, User Attitudes, and Admissions
  - B. An Overworked Computer and Increasing Demands

ABSTRACT

Intergrating Computer Literacy in the K-8 Curriculum

Beverly Hunter, Principal Investigator,
Senior Staff Scientist, Human Resources
Research Organization, 300 N. Washington
Street, Alexandria, Virginia 22314

Many educators recognize the need for schools to make a transition to a computer-literate culture in which skills in information-handling and creative problem solving are central to the learning and teaching processes. Making this transition is a complex process involving staff development, curriculum revision, development and acquisition of new learning materials, and changes in classroom methods.

To aid school districts in planning for this transition, a Guide to Computer Literacy in the K-8 Curriculum was developed. The Guide provides a scope and sequence of objectives from kindergarten through eighth grade, intended to be integrated into the regular curricula in math, science, social studies and language arts.

The Guide was pilot tested in three school districts during the 1981-82 school year.

The two hundred persons who were pilot users of the Guide included classroom teachers, school principals, curriculum supervisors, school district administrators and staff persons with widely varying backgrounds ranging from computer novices to experts in teaching training for computer literacy.

The NECC presentation will provide a summary of findings with regard to integration of computer literacy in the schools and school districts involved in the pilot test.

*This work is supported by the National Science Foundation Development in Science Education program.

ABSTRACT

Your Future with Computers a Career Workshop for High School Students and Their Parents

Susan M. Zgliczynski, PhD, University of
San Diego, School of Education, Alcala
Park, San Diego, CA 92110

This workshop is currently being offered to high school students and their parents in the San Diego area. The course runs for 10 hours on two Sunday afternoons. It is an attempt to provide students and parents with information which will be useful in making future career decisions and it offers an opportunity for students and their parents to get introduced to microcomputer usage.

The first five hours of the workshop is centered around the components of a career decision. Participants are given information about how career decisions are made, what information needs to be gathered about the self and about career opportunities, and when do high school students need to make career decisions. Participants have an opportunity to use a program which runs on an Apple II computer to assess some career interests. The participants are also given specific information about the number of careers currently available which require knowledge of computers, where training in computers may be obtained, and the workshop participants are asked to brainstorm how computer skills might be used in careers which are not traditional linked with computer training. Some examples of career areas not traditional linked with computers are medicine, mental health, law, real estate, business, education, and agriculture.

The second five hours of the workshop is an introduction to microcomputers. Students are paired two to an Apple II Plus System and are taught how to work with this system, how to use software, and how to write a short program in the BASIC language. Examples of software of interest to high school students and their parents is demonstrated. Specific information about further computer training for students and adults in the San Diego area is provided.

Plans are underway to offer a second workshop for adults who are needing similar information to make a change in career and want information about computer-related opportunities.

Tutorial

ADA PROGRAMMING LANGUAGE

Robert Mathis
Old Dominion University

ABSTRACT

ADA is a new programming language developed by the Department of Defense for use in embedded computer systems, but its area of application is much wider. There are already implications that it may replace Fortran, Basic, Pascal, and even Cobol in many situations. This tutorial reviews the ADA progra-ming language and its relevance to teaching computer programming and the use of computers in education.

The first part of the session will be general introduction to ADA. Although the emphasis will be on how the language might be used by people with a small system and a background in only Basic or Fortran (People with background in Pascal would be very well prepared for this session), almost the entire language will be reviewed. This session will review the development of the language and its relevance to programming in the 1980's.

The second part will deal primarily with some larger programs using case studies and emphasizing topdown development, modularity, new types, encapsulation, and concurrent programming. The primary focus of attention during the second session will be on an intelligent terminal program for use on a machine like the TERAK 8519/a.

Teacher Training

COMPUTERS IN EDUCATION:

WHAT'S AN APPROPRIATE WAY TO TEACH EDUCATORS ABOUT THE USES OF COMPUTERS?

by

Lincoln Fetcher
Minnesota Educational Computing Consortium
2520 Broadway Drive
St. Paul, Minnesota 55113
612/376-1119

Introduction

There is a new and exciting awareness in education. Teachers and administrators from all levels of education are coming to realize that computers can be very useful in their teaching and administrative duties. As a result, we at MECC have had increased demands for inservice workshops and classes. We have re-evaluated our orientation toward educating educators and we have created two new courses: one oriented toward classroom use of computers; the other toward use by decision makers. In the process, several issues (not necessarily new ones) have surfaced. After presenting background, I will discuss the issues in terms of our two courses and present my viewpoints.

Background On MECC

The Minnesota Educational Computing Consortium (MECC) is a state organization responsible for providing support services throughout the state relating to computers in education. These services include developing and distributing computer courseware and support materials; evaluating, contracting for and supporting hardware (of several types and sizes); providing inservice training in the form of workshops, presentations and classes; and consulting and coordinating on all matters of computing in education.

The rationale for the Minnesota legislature organizing MECC was to provide equal computer access and support to educators throughout the state. This mandate came in 1973 before instructional computing was widespread, making it easier for organization on a state-wide basis. The result is that most school districts have computers of some form - either microcomputers or timesharing terminals - and that 98% of all students have access to computer aided instruction one or more times during their education. MECC must help educators throughout the state integrate computers into their schools to be used as tools in teaching and administration; ultimately making computers a normal part of life for students. The factors leading to this conclusion are: computers are becoming more and more prevalent; teachers in particular are being presented with students who have already had computer experience and expect to get more of it; teachers need both instructional and administrative support for instructional computing so administrators need to know about computers; and computers are becoming recognized as both instructional and administrative tools.

Previous to fall 1981, MECC's in-service training centered around giving classes and workshops on the more technical and low-level aspects of computers like operating and programming them, and using peripheral devices. This approach was useful (especially learning how to operate computers), but only served part of the needs as we currently perceive them.

Our attitude at MECC about computers in the classroom is that the use of computers as teaching devices should be an integrated part of a curriculum. Computers should not necessarily be used in all curricula, but they should be used only where appropriate and not where they don't enhance an educational experience. In other words, computers shouldn't be used in education just because they exist.

A problem that MECC continues to experience, is our inability to satisfy all requests for workshops and classes so we decided to re-orient our priorities about what should be taught to whom.

Two New Classes

We considered the following:

- everyone interested first needs to become familiar and comfortable with using computers.
- educators do not necessarily need to know how to program (they need other skills first).
- educators not only need to know how to run the physical device, but they need to know what's available in the way of software and how to get it.
- they need to know how and where computing fits into, and how to integrate it into, a curriculum.
- educators must be able to evaluate software once they have found it.
- since educators will not always find the program they need, they should know something about designing courseware.
- educators need to know administrative uses of computers in education.

- there are a variety of courses available from other educational groups that teach programming and more technical aspects of computing.
- we have responsibilities to help educators become involved with using computers, but not necessarily teach them everything there is to know about them.

We then speculated that it would be appropriate to give a course for beginners (who had probably never touched a computer), teaching elementary skills on running the machine, finding and running available software, dealing with hardware and software problems, and evaluating software. That is information everyone needs to know, in one form or another. We actually saw a need for two courses, each with some overlapping materials and some divergent materials. One new courses, called "Using Computers in the Classroom," is oriented toward teachers and using computers as teaching tools. The other, "Instructional Computing Awareness for Decision Makers" is oriented toward administrative aspects of instructional computing and some administrative uses of computers.

The common goals for the two new courses are to familiarize educators with: 1) a microcomputer and how to make it function; 2) elementary technical jargon necessary to understand hardware and software specifications; 3) locating available software from MECC and other sources; 4) how to run a variety of software and recover from common errors; 5) types of instructional applications and methods for presenation; 6) introductory experience in programming. Both classes provide extensive time for hands-on time with computers.

The course for teachers has an intended audience of teachers, and additionally covers (see outline: Figure 1) the areas of : 1) methods of integrating instructional computing into curricula; 2) evaluating the appropriateness and quality of software for the classroom; and 3) planning lessons and designing courses using computer courseware.

The decision makers course (see ouline: Figure 2), intended for school administrators and media center personnel, covers the different areas of : 1) planning for, supporting, and administering computers in a school or district; 2) overview of administrative utility software; and 3) factors in purchasing software and hardware.

Having given these courses a few times, we have had feedback from both participants and instructors. The strongest message that comes back is the need for a lot of hands-on time especially at the beginning. It appears that these classes are appropriate and effective: attendance rate has been high, and participants have said that the courses were just what they were looking for.

Primary Issues and Questions

Some issues have surfaced with the creation of the two new MECC courses, which continue to linger. They are issues that are dealt with differently by everyone and will probably never be "solved", but they are important to acknowledge and talk about nonetheless.

The issues are not easily separated, because they are all inter-related, but I do put them into two groups:

1: What does one need to know to be able to make good use of computers?
  What should be taught? (relating to above)
  How should it be taught? (method, sequence, emphasis)
2: Who should use computers in what way?

Let me present my perspective of the issues and my feelings about them, one group at a time.

The question of "what do you need to know to use computers (appropriately) and therefore what should be taught and how?" seems to generate two viewpoints. One: the practical know-how advocates who say "teach 'em to turn the machine on and off, to run it, to fix simple problems, and what programs are available and where to find them." And in the process, just put people in front of the machine and make them use it. Don't worry about how it works, why, and all that "heady" information. In other words, just teach people the elementary functional things they need to know to run a computer and use software on it. Two: The other viewpoint professes teaching many of the same things necessary to run a computer, but also to teach some of the concepts, processes, and background in instructional computing. The concepts and processes have to do with appropriateness or inappropriateness of computers in a curriculum; strategies of sequence; methods of integrating instructional computing into a curriculum; methods of presentation; how to evaluate software; and what is involved in designing courseware. The background areas include history of computing and instructional computing, terms and jargon, and even concepts of how a computer works, and what programming and programming languages are.

I think either of the two methodes presented above can produce positive results. I tend more towards the concept-oriented school. That is my background and my experience, but I have seen the method of "get them doing it" applied and I consider itno less effective - at least in the short run. My feeling is that it is important to present concepts, processes, and background for two reasons. One: I think the best time to learn about these in the early stages of exposure to a subject. It presents perspective - whether it's integrated right away or not. Two: This may be the only formal course that participants will take on this subject and therefore may be the only chance to be presented with these important aspects. In either case, whether they are immediately learned and integrated or not, these aspects will be useful and important later when more experience is gained and new situations present themselves. Therefore, I would say it is important to begin courses of this nature with lots of hands on time and basic "get 'em going" information, then present theory and background so students leave the course with a rounded beginning The problem with this in tems of the two MECC classes is that 5 two-hour sessions are not enough, for "Computing in the Classroom" especially, I would suggest 7 or 8 sessions.

The second group of issues is "Who should use computers in what way?" and has at least three sub-issues that I have experienced. The first: whether or not professional programmers (those formally trained in programming and computer science) should be the only ones to do "serious" programming. That is, those who are not professional programmers should hire their programming out instead of doing it themselves because they're incapable of doing it "correctly". I get angry about this attitude which seems narrow and selfish. It's not that I don't think good programming is essential or that people taught the concepts of programming and problem solving will not be better programmers (in general) than those who are not. I think everyone should use computers in any way they find appropriate. I'm not a trained mechanic, but I can fix mechanical devices, certainly not with the speed and finesse that a trained and experienced mechanic can, but I don't think I should be restricted from mechanical work. It's the same principal - I think computers and programming them should be for all people.

This may result in some poor quality programs in the market place. It's a phase we must go through and part of the process of society becoming computer literate. Once everyone is brought up with computing as part of their lives, programming will be like writing - everyone does it, some is good, some is bad, and most is okay.

The second issue in this group is "What language should be taught?", one of the more hotly debated questions in the field. I am a Pascal proponent, but I don't think Pascal is the "only" language. I do think Pascal elegantly implements and encourages some extremely important concepts in programming: stepwise refinement and top-down design, modularity, structured programs, user-defined data structures, recursiveness. But, it is not necessarily a language I would present to people who want an immediate notion of what programming is like (participants in our classes, for example). It seems more appropriate to teach a language in which you can do something right away like BASIC or an authoring/teaching language like PILOT or LOGO. These languages illustrate in relatively simple ways the kinds of instructions computers can recognize, and they work well for simple and short tasks. Currently we spend about one hour in each of the two classes programming in BASIC just to give participants some perspective on programming and what is involved in developing useful programs. We use BASIC because it's prevalent, but perhaps it would make more sense to use PILOT or LOGO because there is a closer relationship between what you want the computer to do and how you tell it to do it.

Since we do not teach a language in our two new courses, we are not presented with additional issues of what language to teach in terms of supporting programming concepts, portability of programs, and prevalence of a language (usable elsewhere). These issues are important but beyond the scope of this paper.

Finally, the question "Who should design instructional computing materials?" has come up as an issue at MECC. It is a question because many instructional programs are submitted to MECC for publication that are written by people who know little or nothing about designing instructional materials for computers. Should we teach something about design of this courseware in our classes so teachers will produce better materials, or should we discourage teacher design and programming, and ask them to bring us ideas and we'll do the rest? Again, this goes back to the issue of "who should do what with computers?".

Ideally courseware should be designed by a team: a designer, a programmer, and a subject matter expert (usually the teacher). This provides for several perspectives on the material, a broad base of experience, and a check and balance system to keep the design from going astray of clear objectives. Much of the material sent to us is already designed and programmed by one person. MECC usually makes major revisions and sometimes does rewriting on these intructional programs, and this is done with a team. But, there is also a lot of couseware that's designed, programmed, and sold by one person. Some is good, a lot is bad.

Therefore, it makes sense to me to try to present, in a course, some information on how to design and implement instructional computing courseware. This would then give participants a common base from which to start courseware design. I also think that materials design should not be restricted to an elite group, just as programming should not. And, just as with programming, until we have reached a stage where use of computers in the classroom is as common as blackboards are today, we are likely to go through a period of lower quality courseware materials. Therefore, teaching courseware design will speed up the process of proliferating computers in classrooms and will likely improve the courseware that's produced now, because no matter what is or is not taught, people will continue to design materials on their own.

Conclusion

There is no single way to teach educators about the use of computers, but I do think it is extremely important to teach them as much as possible and to get teachers and administrators involved in computing. Instructors at MECC each teach the two new classes in their own way, and each addresses differently the issues I have brought up. But, at the same time, we have come to recognize the importance of informing educators about more than just the skills of using a computer. Therefore, when teaching introductory classes to educators such as those described here, I would suggest beginning the class by getting participants comfortable with using the machines. Then after this hands-on experience, present some of the concepts concerning instructional computing, and have participants do exercises around the concepts. The most important concepts are about integrating computing into a curriculum, evaluating software, and the processes and considerations in designing instructional courseware. Even if not everything sinks in at the time of presentation, I think the ideas will stick and be useful at a later, more relevant date.

338

Figure 1

"Using Computers in the Classroom"

Class Outline

Session 1

Historical Background On Instructional Computing (10 min)
Operation Of A Personal Computer (35 min)
Hands-On Work With WORD GAME (MECC Elem. Vol. 2) (45 min)
Hands-On Work With MECC Demo Diskette (30 min)

Session 2

Sharing: How Participants Are Using Computer In School (15 min)
Review of MECC Distributed Materials (15 min)
Hands-On Work With Selected MECC Diskettes (60 min)
Copying Diskettes (30 min)

Session 3

Sharing: How Participants Are Using Computer In School (15 min)
How The Computer Works (45 min)
Understanding Commercial Advertisements For Software (15 min)
Evaluation Criteria For Educational Materials In General (15 min)
Group Evaluation Of A Computer Application (30 min)

Session 4

Sharing: How Participants Are Using Computer In School (15 min)
Slide Show: Evaluation Of Software (45 min)
Sources Of Commercial Software (30 min)
Instructional Strategies (30 min)
  Options For Computer-Student Seating Arrangements
  Building An Instructional Sequence
Participants Write A Lesson Plan (30 min)

Session 5

Sharing: Participants' Lesson Plans (15 min)
Choose One: How Computing Fits In The Curriculum (30 min)
OR
Hands-On Work With Commercial Software (30 min)
Choose One: Demonstration Of Peripherals (60 min)
OR
The Idea Of Programming (60 min)
Information Resources And Future Directions (15 min)

(MECC-DEC81)

Figure 2

"Instructional Computing Awareness for Educational Decision Makers"

Class Outline

Session 1

Background On Instructional Computing (10 min)
Computing's Place In The School Program (20 min)
Operation Of a Personal Computer (30 min)
Hands-On Work With MECC Courseware (60 min)

Session 2

Types of Instructional Applications (15 min)
Review of Courseware Sources (15 min)
Copying Procedures (20 min)
More Hands-On Work With MECC and Commercial Courseware (30 min)
The Idea Of Programming (40 min)

Session 3

Overview Of Administrative Utility Software (15 min)
An Electronic Worksheet: Visicalc Demonstration (30 min)
A DataBase Package: DB Master Demonstration (30 min)
Word Processing Packages: Applewriter, Word Star, Magic Window Demonstrations (45 min)

Session 4

Hands-On Work With Administrative Utility Software
  Visicalc Exercise (60 min)
  Try Out DB Master and A Word Processing Package (60 min)

Session 5

Operation of MECC Timeshare system (15 min)
Review of Major MTS Applications (45 min)
  SEARCH Demonstration
  MINITAB Demonstration
  MAIL Demonstration
Personal Computer Hardware Purchasing Factors (15 min)
Personal Computer Software Purchasing Factors (15 min)
Planning a District Computing Budget (10 min)
Information Resources (10 min)
Future Directions (10 min)

(MECC-DEC81)

340

COMPUTER LITERACY FOR TEACHERS:
A District Do-It-Yourself Kit

Judith A. Hopper
Glenn Bruckhart

Arapahoe County Schools #6
Littleton, Colorado 80122

Computers are now an everyday fact of life in business, government, and industry. School districts are also commonly using computers for payroll, accounting inventory, and student schedules and records. In fact, a survey done in 1980 by Al Bork and Jack Chambers found that 90% of the school districts in this country use computers in some way.

But computers have still not had a significant impact on the curriculum in most schools. Teachers are not using them as an instructional tool because they do not know how and often are not aware of their instructional potential. Widespread incorporation of the computer as an instructional tool is certainly a function of the general computer awareness and literacy of teachers. It is therefore essential that training in computer literacy be provided and made attractive and usable to large numbers of staff members in any school district beginning an instructional computing program. Such a training program needs to be a face-to-face, non-threatening, functional, and continuing. It is believed this can best be done for any school district by people within that district.

What follows is a description of how one school district in Littleton, Colorado is meeting the challenge of effective teacher literacy training. The program was spearheaded by a small group of teachers who had some background in computing and a strong interest in pioneering such a program.

The computer literacy program for staff members contained three distinct phases: initial awareness, hands-on introduction, and a computer literacy inservice. These three parts met the district philosophy, i.e. use of computers within all disciplines, and mass student exposure and use of the machines.

For Phase One, initial awareness, a small-scale computer fair was held. A number of local vendors were asked to supply the hardware, relevant software, and a representative to answer questions. In addition, several in-district teachers agreed to be available to aid in demonstrations and encourage their colleagues to experiment with the machines and courseware.

The entire secondary staff was invited to come on a drop-in basis, walk through, look, and interact with the microcomputers on display. The atmosphere was low key, friendly, and non-threatening.

The fair proved effective in bringing a large number of teachers in contact with a microcomputer, many for the first time. It also made those teachers aware of at least a few of the possible uses of microcomputers in the classroom.

Each secondary school was asked to designate one interested staff member as a "computer coach." This person became responsible for the introduction of the computers into his own school and their day to day use and maintenance. It therefore was the "computer coach" who implemented Phase Two of the program in his respective school.

The coaches were encouraged to have each department meet with them in the computer room. This afforded the opportunity for each department's members to gain some hands-on experience. Relevant courseware was provided and the department members could discuss how it actually might be used within their curriculum.

The meetings also established the coach as the school's in-house consultant to help them with computer related problems. The coaches had previous programming skills, catalogs of available software, and enthusiasm to help get teachers started using the computer. They were also urged to identify a pool of competent students to act as computer aides for any teachers desiring classroom assistance.

Phase Three of the literacy program was a one semester hour inservice for all interested secondary staff. The class was held at a centrally located high school and has been offered both during the summer and as an evening course during the school year. Sixteen microcomputers with a networking system were used for the classwork with an alternate model available for demonstrations.

The fifteen class hours were broken into six two and one-half hour sessions. The main goal throughout the six sessions was to help teachers become literate and to be comfortable with the computer. Additional objectives were familiarization of teachers with:

1. Computer vocabulary
2. Existing software
3. Construction of simple programs
4. Capabilities and limitations of computers
5. Computer classroom applications
6. Software sources

By the end of the course the teachers were able to write some simple programs of their own. They could do some editing of an existing program to make it more useful for their own purposes. They became familiar with some sources of additional software. They were aware of and exposed to both CAI and CMI functions of the machine. And they could see the relationship of the computer to instruction and the part it will undoubtedly be playing in the future.

Thus far about 30% of the secondary teachers in the district have completed all three phases. Every time the inservice has been offered the classes have been full to overflowing. Additional advanced programming classes have also been requested. It is anticipated Phase Three will be offered on a continuing basis for as long as there is interest. Ideally all staff should complete the program and work is continuing to get as close to this goal as possible.

As a check on the progress of the participants who have completed the three phases, a follow-up questionnaire was distributed. The compiled results appear on the following page.

Comments accompanying the questionnaire indicated that lack of preparation time was the major factor limiting one's use of the computer. However, virtually all respondents indicated they would like to make greater use of the computer. This indicates the need for additional planning time for initial implementations.

It is also vital that district administrators receive computer literacy training because they approve all requests for the literacy programs themselves as well as the corresponding computer hardware. Therefore, a special five session seminar was held for all of the administrative staff, from the assistant superintendents down to the assistant principals.

There are a number of advantages to using district equipment and personnel to get one's own computer revolution started. Teachers instructing teachers from within their own district are less threatening than instruction from an "outside" source. Participants are generally more relaxed in a peer-to-peer relationship and not as reluctant to ask questions and say "I don't understand."

The in-district instructors know the equipment and can be a guiding force through the intricacies of the machine. The participants actually learn programming and use of software on their own machines. The knowledge is one-hundred percent transferable.

A low-key approach proved to be a good instructional method. The teachers are there to learn to coexist with and not be threatened by the computer. They do not have to worry about completing projects, taking midterm exams, studying for finals, or reading required text materials. Although university credit cannot be earned, recertification and salary schedule credit can. Also the course and material covered is structured to one's own district's philosophy and goals.

Because the course is taught through the district, it can be tailored to each participant and their particular school needs. Also, a significant advantage to in-district instruction is that the core instructional team is available on a day-to-day basis for problem solving and consultation.

In summary, the three-phase in-district program outlined met the four criteria established for such a teacher training program. First, there was face-to-face instruction. Each of the three phases provided the participants with exposure to the machines and someone to answer questions, help, and instruct. Most teachers will neither become nor remain computer users if they must develop the expertise on their own.

Secondly, the situations were non-threatening. For those with no knowledge of computers, the machines can be a mysterious device shrouded in technical jargon. There is a feeling among teachers that only those people trained in mathematics and science have the capabilities required to use a computer effectively. Everything possible was done to dispel this myth and make each participant as comfortable as possible with the machine. The point was made that computers are just that, machines!

Thirdly, the information and instruction given was functional. Throughout the six sessions of the inservice all materials and programs presented were geared to education and how a teacher might make use of the computers with his classes. Problem

QUESTION	RESPONSE		
	Never	Occasionally	Often
1. I have used the computers:	37%	50%	13%
2. I have asked for help from my computer coordinator:	50%	41%	9%
3. I have received help from my computer coordinator:	38%	49%	13%
4. I have used information from the inservice:	12%	69%	19%
5. Since taking the course I am more likely to read articles on computers:	13%	51%	36%
6. I would use the computers if I had software readily available:	6%	54%	40%
7. Since taking the course I have found it easier to understand literature containing computer terminology:	10%	42%	48%

solving, tutorials, CAI programs, grading software, and simulations were written, discussed, and demonstrated with respect to their use in the classroom. Participants needed applications they could take back to their school and classrooms and use.

Finally, the program is continuous. With the instructors in the district, they are available for consulting with those who have completed the program and also to continue the program for as long as it is needed. The inservice phase has now been given seven times for district personnel. Also an advanced level can be added for those desiring more programming instruction and technique.

This literacy program was given partially during the school year. The small scale computer fair was held after school, as were the hands-on introductions in the individual schools for each department. The inservice phase was given in a three day summer session and also taught one night a week for six weeks during the school year. The summer sessions were the most productive because the teachers were free from their pedagogical responsibilities and could give their full concentration to the course.

All school districts can implement a similar program. The basic tools needed are: A dedicated team of interested teachers with some computing experience, a cooperative administration, and willing participants. These three tools and the model described above will enable you to "DO IT YOURSELF."

COMPUTER LITERACY FOR TEACHERS: AN INTENSIVE PROGRAM

Edward L. Vockell
Robert H. Rivers
Diane Kozubal
Purdue University Calumet
2233-171 Street
Hammond, Indiana 46323
(219)844-0520, Ext. 362

To an objective observer, the illiteracy of our current generation of teachers with regard to the computer would have to be regarded as simply astonishing. Recently (in 1980), we took a survey of the beginning graduate students in some of our courses at Purdue University Calumet. We discovered that among these beginning in-service teachers over 90% had completed their entire undergraduate program without ever taking a course which required them to do anything whatsoever at a computer terminal. Indeed, it seems likely that teachers who completed their undergraduate training earlier than this sample of teachers probably have had even less exposure to the computer.

This absence of exposure to the computer is critical for two reasons. First, it seems obvious that such teachers cannot be expected to take full advantage of computer technology in their schools and classrooms. Secondly, and even more critically, these teachers are responsible for raising a generation of children in whose lives the computer will play a much more important role than it will ever play in the lives of these teachers themselves. For these reasons, it is obvious that we need to provide teachers with a greater degree of computer literacy.

Several contemporary teacher-educators have acknowledged this need for increased computer sophistication among teachers. In recent years, there have been an increasing number of graduate or in-service programs designed to develop computer literacy among teachers. Such courses typically follow a pattern like this: (1) An instructor lectures to the students about such topics as What the Computer is Capable of Doing, The History of Computing, and the Ethics of Applying the Computer in Modern Society. (2) The instructor shows the students various types of hardware and software currently available for educational applications. (3) Vendors of various computer materials often make guest appearances to show the students what their companies have to offer for educational purposes. (4) Students do some programming in a computer language such as Basic.

THE PURDUE PROGRAM

At Purdue University Calumet we have followed a different approach to promoting computer literacy among teachers. Taylor (1980) has described three separate rules which the computer can play in education; the tutor, the tool, and the tutee. In our program at Purdue we have employed the computer largely as a tutee---that is, our students have been required to teach the computer something as their primary activity during the course. Taylor suggests (and we believe) that learners (especially adult learners) can learn a great deal about the capabilities of the computer when they are required to teach the computer to actually do something.

For the past five years we have offered the course on Computer Assisted Instruction on a four week intensive basis during the summer session. We have chosen this time partly because students like the intensive nature of the course and partly because during this time period the computers are more readily available than they are at other times during the school year. About 75% of the students have been beginning-level teachers, taking the course as an elective towards their masters degree in education. The other 25% has consisted of guidance counselors, school administrators, and a few college teachers.

The course consists of 45 hours of scheduled contact time, but only 3 of these hours consist of lectures. The lectures are staggered in such a way that some students are working on computer terminals while other students are attending the lectures. During the first class session the students receive a one-hour lecture and demonstration on how to use the computer. During this introductory session, they are shown how to turn the computer off and on, how to load and run programs, and how to perform simple operations necessary to run the programs which are already available on the computer. They are also shown the textbooks (Dwyer and Kaufman, 1977, and Doerr, 1979) and are told how these fit in with the course requirements. At this time the instructors also explain to the students when they will be available and what the course requirements will be to achieve various grades. During each of the following two weeks, the students also receive an additional one hour lecture during which variour programming topics are covered such as how to write loops in programming, how to use the Edit Mode, and how to use graphics effectively. During these sessions questions raised by students are also answered.

Specific performances are required to obtain various grades. For example, to get an A in the course, a student must do the following; (1) Run at least 25 of the educational programs which are already available on our computer. (We have a file of approximately 200 games, drills, and simulations). (2) Explain the logic of at least 2 simulations to another student or to one of the instructors. (3) Modify 3 existing simulations in some significant fashion. (An example of such a modification would be altering the program from our Prime minicomputer to operate on our TRS-80 microcomputer). (4) Write 2 computer programs at a designated level of sophistication. (5) Make a plan to write another program at a higher level of sophistication. These requirements are rigorous but attainable. We have never had a student claim that the course was too easy, nor do we find many students taking the course for any easy grade.

The students work at either TRS-80 or Prime computer terminals to perform their assignments. During the first couple days, most of the students merely run existing programs, but by the end of the first week many students are already writing their own programs. By the end of the third week, all the students are programming at a reasonable level of sophistication. The output from the course has covered a wide range of topics, including the following examples:(1) a drill on Spanish vocabulary and sentence structure; (2) a drill on the Dewey Decimal System, which provides a different set of books for each student to classify as he or she interacts with the program; (3) a program to teach Morse code by requiring the student to translate riddles written in the code; (4) programs to provide guidance to students in course selection; (5) a program to simulate buying and selling processes in a large department store.

Reaction to the course has been quite favorable, and student enrollment has increased over the five-year period, even though the course has never been required on anyone's plan of studies. The major criticisms of the course have been that we do not have enough terminals to allow the students as much time as they would like to spend at the terminals (most students do far more work than what may be required to meet the course requirements) and that we are not able to take as many students into the course as would like to enroll.

THE EVALUATION

Since our students work quite hard during the course, we are reluctant to add to their burden by having them fill out a large number of questionnaires or tests while taking the course. However, we feel obligated to evaluate the effectiveness of this procedure for instilling computer literacy among teachers. Rather than conducting a massive study with any single class, therefore, we have conducted "mini studies" during each of the past three years. Two years ago, we conducted a survey (Vockell and Rivers, 1979) in which we collected follow-up data on students who had taken the course either one or two years previously to see what they had done with the information they had acquired during the course. Last year, we administered tests to our CAI students and to a control group to see whether such a program enhanced critical thinking and creativity scores among our students (Vockell, Rivers, and Kozubal, 1981). This year, we administered questionnaires to determine whether attitudes toward computer usage changed as a result of the course and whether useful programming skills could be instilled through such a course. A summary and integration of these three sets of data will provide a useful means of evaluating the effectiveness of the course.

The follow-up survey (Vockell and Rivers, 1979) indicated that one or two years after completing the course almost none of the students had access to a computer in their classrooms. In spite of this absence of computers, the students were nevertheless glad they had taken the course and they cited specific examples of why the course was beneficial. The two most cited benefits were that the course had enabled them to think about educational problems in a new and more useful perspective and that the course enabled them to understand more fully how the computer could and could not help them as educators. Note that these enefits occurred as outcomes of a course in which no attempt whatsoever was made to lecture on the very topics that the students found to be the most beneficial outcomes. The students had merely learned to program effectively. They had taught the computer to solve educational problems, and by doing this they themselves had learned to think differently, and they had learned what could and could not be done with the computer in education.

Next, we examined what might be called a very interesting and important by-product of our course. We were certain that our course taught useful programming skills, and we had acquired evidence that the students who took the course saw benefits in their own educational careers. Now we wondered whether or not we could find actual evidence of a change in thinking style as a result of taking such a course. We administered the Watson-Glaser Test of Critical Thinking and the Torrence Test of Creative Thinking at the beginning and at the end of the course. We administered the same tests to the students who were enrolled during the same semester in a highly-regarded educational psychology course. The analysis of the data (Vockell, Rivers, and Kozubal, 1981) provided clear evidence that the course did enhance critical thinking skills. Specifically, the students who completed the Computer Assisted Instruction course showed a significant improvement in deductive reasoning over students who had not taken the course. (They showed similar but non-significant improvements on each of the other Watson-Glaser subtests as well). The results with regard to creativity were clouded by a "ceiling effect." It appears, however, that the CAI students also showed an improvement in creative thinking compared to the non-CAI students.

These critical thinking and creativity results are quite impressive. It seems obvious that the ability to think creatively and critically would be a significant benefit for teachers. However, these topics do not appear to be taught effectively in many undergraduate and graduate programs. If a class which teaches effective computer programming skills and computer literacy can as a by-product develop more effective critical thinking and creative thinking skills, this provides an additional incentive for encouraging schools of education to offer such courses for their students.

Most recently, we have asked the students to evaluate themselves with regard to specific objectives related to computer literacy. These objectives are taken from Taylor, Poirot,& Powell (1980), and they represent a consensus of what experts feel would indicate computer literacy among teachers. The seven competencies are stated in Table 1 in exactly the format in which they appeared on the questionnaire given to the students. The respondents rated each competency on a 10-point scale (1 = not competent at all, 10 = very competent). The first number after each objective in Table 1 indicates the mean pretest score on each competency. The second number after each competency indicates the mean posttest performance of the same students. In each of these seven cases, the difference between pretest and posttest performance was very highly statistically significant. The number in parentheses after each competency indicates mean performance on the same scale attained by the students in the Carroll and Johnson (1981) study. The Carroll and Johnson comparison is included here because their course (which involved much less emphasis on programming skills) was designed specifically to help teachers attain the goals specified in Table 1.

Table 1 actually represents a somewhat conservative estimate of student growth with regard to some of the competencies. This is because several of the students in the course to which this questionnaire was administered already possessed some of the competencies at the beginning of the course,and therefore there was a "ceiling" limiting how much students could improve on such a scale. Table 2 solves this problem by focusing entirely on those who answered either 1 or 2 on the pretest (that is, it focuses on the "truly illiterate"). These results indicate an even stronger pattern of improvement than was seen in Table 1.

Of course, all these results are based on the self-perceptions of the respondents. However, their is no compelling reason to believe that such self-perceptions are inaccurate. At any rate, Tables 1 and 2 indicate that a large number of persons who felt extremely incompetent at the beginning of the course, felt that

Table 1 Mean pretest and posttest ratings of computer-related competencies (1= Not competent at all; 10 = Very competent)

	Pretest	Posttest	Carroll & Johnson
1. I can read and write simple programs.	1.80	7.75	(7.1)
2. I have had experience using educational software	2.00	7.63	(6.9)
3. I know some basic computer terminology	2.58	7.96	(7.3)
4. I understand what types of problems can and cannot be solved by computers.	2.19	6.81	(5.9)
5. I can make use of educational computing information	2.52	7.59	(7.1)
6. I am able to discuss the history of computing	1.56	5.85	(7.1)
7. I am able to discuss moral issues related to societal uses of the computer	1.91	6.06	(7.4)

Table 2 Posttest ratings of competencies by persons who were "incompetent" at the beginning of the course. (Item numbers = competencies in Table 1.)

	RESPONSES										
QUESTIONS	1	2	3	4	5	6	7	8	9	10	Pct. 8 or above
1	0	0	0	0	1	4	0	3	6	10	79%
2	0	0	0	0	2	3	3	4	7	8	73%
3	0	0	0	0	0	3	2	3	5	5	72%
4	0	0	2	2	3	3	5	4	4	2	40%
5	0	0	0	0	3	1	4	6	5	4	65%
6	0	1	3	2	10	1	4	6	3	1	32%
7	0	2	2	2	6	2	3	6	3	3	41%

they had developed a high degree of competency in many important skills by the end of the course.

Table 3 summarizes the students' self-perceptions with regard to specific computer programming skills. The skills listed in this part of the questionnaire cover a wide range of areas in which it would be possible to use the computer in the classroom. Most of the programs written by the students during the course fit into the categories covered by the first three questions. Nevertheless, the students displayed highly significant pretest gains on all ten skills. The most obvious explanation for this high degree of generalization is that students who practiced their skills by writing relatively simple programs easily perceived how these skills could be applied to other situations which might come to their attention in the future.

Table 4 shows how the attitudes of the students shifted with regard to specific ways in which the computer can be applied to educational settings. The most substantial changes occurred in a greater willingness to use the computer to individualize guidance and scheduling for students at all grade levels. Likewise, there was a significant increase in the willingness to use the computer for problem-solving practice and to "fool around" with computers at the high school and college level. In several cases, (e.g. library retrieval) there were no significant gains because the students already endorsed these applications at the beginning of the course.

A disappointing result found in Table 4 was that we did not find significantly greater willingness at the elementary or middle-school level to let children use the computer to solve problems or to "fool

Table 3 Mean self-perceptions of ability to perform specific educational tasks with help of a computer (1 = certainly could do it"; 4 = certainly could not".)

	Pretest	Posttest
1. The computer will provide students with drill and practice over factual material covered in a course you teach.	2.78	1.11
2. The computer will provide students with drill and practice on factual material or simple problem solving exercises in such a way that each run through the program is likely to differ from other runs.	2.97	1.29
3. The computer will provide the students with academic, technical, or career information in such a way as to enable them to be aware of options and make appropriate choices.	3.09	1.71
4. The computer will schedule courses and instructors in a high school or college program.	3.33	2.36
5. The computer will provide a simulation of a real-life problem setting.	3.28	2.00
6. The computer will play a game which you understand (such as chess, bridge, rummy, or dice) with a student.	3.19	1.61
7. The computer will keep records of student performance.	2.83	1.43
8. The computer will keep a checkbook record of deposits, expenditures, and balances.	2.91	1.65
9. The computer will project future trends in school populations based on designated assumptions.	3.31	2.60
10. The computer will test students and keep a record of their performance.	2.04	1.46

Table 4 Mean ratings of how often the computer should be used for tasks at designated grade levels (1 = very often, 4 = rarely or never)

	Elementary (K-6)	Junior High (7-9)	High School (10-12)	College
1. Drill & practice on terms or concepts.	2.05 1.94 NS	2.08 1.69 .009	2.22 1.72 .001	2.40 2.26 NS
2. Problem-solving practice	2.43 2.37 NS	2.03 1.83 NS	1.97 1.60 .02	2.08 1.56 .006
3. Simulation of real-life problems which would otherwise be difficult to do in the classroom.	2.89 2.63 NS	2.34 2.03 .05	1.89 1.78 NS	1.74 1.66 NS
4. Computer programming by students to solve problems.	3.20 3.29 NS	2.46 2.54 NS	1.88 1.83 NS	1.54 1.43 NS
5. Computer programming to "fool around"(explore out-comes of computer usage).	2.91 2.69 NS	2.39 2.17 NS	2.19 1.81 .03	2.02 1.53 .02
6. Provide data from libraries and similar sources.	2.57 2.14 .01	2.06 1.86 NS	1.53 1.50 NS	1.46 1.40 NS
7. Structured guidance or counseling for students.	3.21 2.94 .10	2.71 2.40 .10	2.37 1.77 .003	2.43 2.00 .06
8. Scheduling of programs and courses for students.	3.00 2.46 .006	2.17 1.74 .04	1.75 1.33 .02	1.60 1.17 .009
9. Testing of Students.	2.71 2.60 NS	2.50 2.31 NS	2.25 2.14 NS	2.29 2.14 NS

around." Our students saw these applications as valuable for high school and college students, but not for younger children. It is our feeling, however, that young children can benefit from such uses of the computer and we would like to pass this conviction on to our students. It seems likely that we can accomplish this by focusing some specific attention on examples of programs which encourage young children themselves to program the computer.

CONCLUSIONS AND SUMMARY

Our research has shown that when inservice teachers develop computer programming skills by actually writing programs for educational use, they both (1) learn important basic programming skills and (2) develop computer literacy. We feel that this "total immersion"approach is the most effective way to prepare teachers for their role as users of computer technology and as guides to introduce their students to the world of computers. As a very important by-product, the teachers learn to think more critically and creatively and feel they have access to better strategies for solving educational problems.

The ideal CAI course for teachers would probably follow a format like the one described in this report, but would also (1) focus on some specific information about computer history, (2) discuss some of the ethical issues in computer applications, and (3) demonstrate some examples of programs to encourage young children to use the computer to solve problems creatively. Such changes could be made without detracting from the applied programing emphasis of the course. Such a course would help teachers develop a solid computer literacy.

REFERENCES

Carroll, T.,& Johnson, N. Educating urban elementary teachers in computer science. Proceedings of NECC 1981, North Texas State University, Denton, Texas, June, 1981. 302-308.

Doerr, C. Microcomputers and the Three R's: A Guide For Teachers. Rochelle Park, N.J.: Hayden, 1979.

Dwyer, T.A., and Kaufman, M.S. Computer Programming in BASIC for Everyone. Fort Worth, Texas: Radio Shack, 1977.

Taylor, R.P. (Ed) The Computer in the School: Tutor, Tool, Tutee. New York: Teacher's College Press, 1980.

Taylor, R.P., Poirot, J., and Powell, J. Computing competencies for school teachers. Proceedings of NECC/2,1980, 130-136

Vockell, E.L., and Rivers, R. A computer simulations course for inservice teachers. The Computing Teacher, 1979, 7, (2), 53-55

Vockell, E.L., Rivers, R., and Kozubal,D. A CAI course for teachers: Stimulus to creative and critical thinking. The Computing Teacher. 1981, 9 (4), 35-39.

Pre-College Computer Science

Problem Solving, Teamwork, and Structured Programming -- in High School?
The Story of the High School Computer Science Project

W. E. Baird
Computer Science Department
The University of Tennessee - Knoxville, TN 37996-1301
(615) 974-5067

ABSTRACT

A two-semester course for high school and junior college use teaches programming in UCSD Pascal with interactive color graphics on the Apple II microcomputer. High-level animation graphics, background and foreground scenes, a scrolling text window which displays program text as it is executed, and single-keystroke command input are among the features of the Rascal/INTERPAS (c) software. The courseware for the first semester was funded by the National Science Foundation. This first-semester course is now available through Gregg/McGraw-Hill.

A second semester is being written by five authors from both high school and university environments. First testing will begin in January 1982, with publication scheduled for early 1983. The team approach to programming as problem solving is emphasized. The course capitalizes on limited machine access to reinforce teamwork and off-line coding and debugging. Pascal's structured nature facilitates the breakdown of large programming tasks into sub-tasks. These are written and tested by the student teams before merger into a complete master program at the end of each lesson.

THE ISSUES

1) Can we provide structured programming skills in a pre-college environment, given limited teacher background, high student-to-machine ratios, and students who currently seem more interested in interactive color video games than in academic achievement?

2) Can students learn to work together in teams on large programming tasks, breaking such tasks down into smaller parts for coding and testing before assembly into a working master program? Can they learn to use off-line time wisely?

3) Can students acquire debugging skills which generalize into non-egocentric ways of looking at personal mistakes?

High School Computer Science - A Curriculum Project for the 80's

The use of interactive color graphics has proven to be a powerful motivator for youth as witnessed by the exploding growth of video arcade games. But back in the late 1970's few were thinking of the potential of exisitng microcomputers to stimulate greater student interest in learning to program. Initial computer encounters by students were textually oriented and could be quite intimidating. Teachers lacking previous experience with computers were mostly staying away in droves. Back "in those days" learning to program meant learning Basic since it was widely implemented on micros, was an interpreted language (which meant easier error trapping), and was supposed to be easy to learn.

Those who wrote educational software were mostly teachers who had taught themselves Basic and those who had left the classroom to write programs. Despite scattered quality, typical programs were long, entered from hard-copy text, poorly documented, and undecipherable by most who wished to modify them for local classroom or hardware environments. Courseware, if it existed at all, consisted of books on how to teach yourself Basic with example programs from the world of mathematics and business. Student motivation was highest among those with strong academic backgrounds in math or clear vocational goals in data processing. The state of the art in graphics was Hangman.

The motivational value of color graphics is now as clear as Saturday morning cartoons and video arcade games. The Apple II was one of the first common micros to implement high-resolution graphics. Apple also supported Pascal through addition of a single card to the motherboard. It thus became the machine of choice. The next questions became: "could beginners learn Pascal - a compiler language?" And: "could the 'momentum' of Basic as an in-place language be overcome in schools which started and grew with this language of the '60s?" The answer to the former question seems clearly to be yes. On the second question the jury is still out.

The need for teaching structured programming early has been discussed elsewhere (1). Students who learn to write code in Basic have to be "reconditioned" later in order to approach large, complex programming tasks and break out smaller sub-tasks for coding. Reading a Basic program is like trying to unravel a mass of tangled string. Basic is no longer the only language implemented on microcomputers. Schools are rapidly buying computer hardware and looking for academically credible ways to use the new micros. The need for courses which guide teachers as well as students through the day-by-day acquisition of language tools while providing a sense of historical perspective on how we got to this point with computer technology is immediate.

So it seems logical to use the motivational engine of color graphics to pull the academic train of needs for (1) systematic approaches to problem solving; (2) programming fluency in a widely used high-level language; and (3) broader knowledge of computer science. This can be done by teaching students to program their own graphics activities (including games) on a school microcomputer. By means of the structured teaching language of Pascal, problems of growing complexity are introduced which eventually require breakdown into smaller sub-tasks and team programming to reach a solution. Guided exercises are used to teach new language tools such as variable declaration, control structures, how to use procedures and pass parameters, one- and two-dimensional arrays, string manipulation, and other language tools. Figure #1 is the Table of Contents for the Teachers Manual of the First Semester.

A previous NECC paper (2) has described how the High School Computer Science Project looks to the user. However, little was said in that paper about the reasons for the current design and the implications for the future of pre-college computer science. Through a grant from the National Science Foundation, a team of University of Tennessee faculty, high school teachers and graduate computer science students compiled an 18-week introductory course for high school students which teaches structured Pascal programming by means of an interpreter known as INTERPAS. This interpreter conforms with standard UCSD Pascal. It features a graphics window with scrolling program text at the bottom, single-keypress command entry, an immediate mode for single-line execution, clear error messages and editor/filer interaction at all times. The course has been tested through three iterations in public schools, resulting in many "user friendly" modifications necessary in the school environment.

OBJECTIVES

The first 18 weeks of the course contain a number of reasonably sophisticated programming concepts up to the level of procedures with variable parameters, but one of the prime objectives of the package is to dispel computer anxiety on the part of both students and novice teachers. Written for the student of average ability, it contains few mathematics prerequisites. Yet it adheres closely to the recommendations of the ACM Computer Science Task Group for a one-year course (3). As that study group recommended, HSCS provides:

1) practice in making appropriate use of computers as tools for problem solving in a variety of circumstances;

2) realistic concepts of the power, usefulness, and limitations of computers;

3) knowledge of the role of computers in current information processing and the effect on social structures of the application of computers;

4) a context from which to consider possible future directions in computing.

Many of the study topics recommended for a high school course by that ACM Task Group are contained in the computer literacy readings and the programming exercises and assignments of the HSCS course.

Students work in teams of three, with a maximum student-to-machine ratio of fifteen. Copy masters describe the programming assignment and new language features in each lesson. These are duplicated and handed out at the start of each lesson-week to prevent stratification of the class by ability. Higher ability students are encouraged to work on "Blue-Sky Projects" with each lesson, thus permitting horizonal but not vertical enrichment. Despite mixing of 10th, 11th and 12th grade students with a wide variety of academic records, the teams have worked together quite well in the test schools. If team "decay" occurs, re-grouping usually solves the problem. Machine access is controlled by a timer device, but student use of time for off-line coding and debugging is not very efficient. Peer pressure and the team approach to solving each problem seems to indicate good rentention of new language concepts, as indicated by continued use of such tools in subsequent lessons. University computer science faculty visiting the test schools have expressed surprise at seeing high school students using structured programming techniques.

After two years of formative testing in a variety of high schools, the first semester of HSCS is now ready for distribution by a commercial publisher. It will consist of seven diskettes containing the INTERPAS interpreter, graphics editors for figures and scenes, a utility package which will permit conversion to standard UCSD Pascal for compiling, numerous example programs, including solutions to quizzes and the Blue-Sky Projects, etc. These diskettes are contained in a Teachers Manual along with seven tutorials for the beginning teacher describing requirements for teaching the course, equipment, INTERPAS, Filer, Editor, disk management, and troubleshooting.

The student book contains readings in "computer culture" - the history, uses, and implications of computers for society.

A license and special keychip protect the software.

A formal summative evaluation of the first semester students is now underway in nine high schools, with results expected in June 1982.

THE SECOND SEMESTER:

The need of most schools to install curricular offerings into 36-week, two semester schedules has been a principal reason for designing and testing a second semester of the course described above. Other reasons include a need for greater depth of programming skills, more vocational emphasis than the game-centered, graphics-based first semester, and the inclusion of details which simply would not fit into the first semester.

This authoring task is being undertaken by a team of five professionals, with backgrounds in university computer science, learning psychology, high school math and science, and programming. Each lesson part will undergo two revisions before testing in two high school classrooms with students who completed the first semester.

Figure #2 is the current Table of Contents for the second semester. The objectives are both more limited and more ambitious than those of the first

350

semester. The goals are:

1) to guide students toward an ability to redefine assigned tasks into smaller, more easily solved sub-tasks for solution by coding;

2) to provide support for teamwork approaches to goal #1 above, with team size growing along with problem complexity;

3) to provide readings for background understanding of the "real world" of computers in science, business, and industry;

4) to produce students who can write, test, correct, and implement readable code in Pascal, modify it as needed, and stand behind it with pride.

These are rather ambitious goals for a first course with average students. We feel that they can be met.

The second semester provides more extensive programming projects for students than the first. Twelve projects are suggested, with example solution programs for each suggested sub-task provided on a disk for the teacher. The projects evolve from graphics-based management of animals boarding an ark, through text and string manipulation on the video monitor, to formatting of hard-copy output to a printer. After ten weeks of "rehearsals" with pre-defined projects, the entire class seeks out a programming problem in the local school environment, contracts to produce a solution for the local "customer", and spends two weeks producing the working code. The final team project is either a version of an "Adventure" game or a mailing list program for business form letters.

The INTERPAS interpreter is abandoned after three weeks of the second semester as students are introduced to compiling. This leads to an increased need for machine access. In most secondary environments, such access is limited. The second semester will require at least one printer and one dual-disk machine to alleviate part of this problem by encouraging off-line debugging and permitting faster compilation. Students in the test schools generally made inefficient use of off-line time in the absence of careful teacher supervision.

A prime goal of the second semester is to take the first-year programmer to a fairly high level of functioning. The final project will require the integration of sophisticated language tools, clear problem definition, and the ability to work on sub-tasks in concert with other programmers toward a common goal. Are these not the requirements of a good programmer in almost any jòb?

IMPLICATIONS

The result of three years of experience with structured programming in the high school classroom implies the following for the future of pre-college computer science education:

1) It is possible to teach logical, structured programming to students of average ability with the use of color graphics as a motivating agent.

2) Students can learn to break down larger problems into smaller ones for solution, and test solutions to such smaller procedures before combining them into a master solution.

3) Working in teams can make the limited machine access in most high schools into an asset as peer pressure and sharing of math skills produce teams which are better problem-solving units than individuals.

4) Debugging programs can help students, particularly minorities and females feel less threatened by errors as "failure indicators" and more aware that mistakes are nuisances which plague the work of all programmers.

5) Off-line coding and debugging is not always facilitated by limited machine access, but is crucial as a learned skill. More thought must be given to how to encourage better student use of time.

For information about Computer Power, contact

Eric Hawks, Product Manager
Gregg/McGraw-Hill
1221 Avenue of the Americas
New York, NY 10020

REFERENCES

(1) THE ELEMENTS OF PROGRAMMING STYLE: Kernighan, B. W. and Plauger, P. J.; McGraw-Hill Book Co.; New York, 1974.

(2) STRUCTURED GAMING: PLAY AND WORK IN HIGH SCHOOL COMPUTER SCIENCE by J. M. Moshell, G. W. Amann and W. E. Baird; NECC/2 Proceedings, June, 1980; pp. 266-270.

(3) COMPUTER SCIENCE IN SECONDARY SCHOOLS: RECOMMENDATIONS FOR A ONE-YEAR COURSE by Jean B. Rogers and Dick Austing; from TOPICS, Computer Education for Elementary and Secondary Schools; January, 1981; pp. 48-53.

FIGURE 1

LESSON	TITLE OR NEW LANGUAGE FEATURES	PROJECT
1	Quilting - production of lo-res tesselation patterns	Getting started; patterns from seeds
2	Immediate mode and Turtlegraphics	First verb commands
3	Cartoon animation of figures	More command verbs
4	Introduction to complete programs	Using the editor
5	Variables and constants	Learning coordination
6	Procedures	Stage show
7	Procedures with Parameters	The Big Show
8	Loops	Control structures
9	Loops within loops, boolean variables, and complex conditions	Pre-defined conditions
10	IF (they liked it) THEN (take a bow) ELSE (get off the stage)	More control structures
11	Data types, arrays, FOR statements	Shelving things
12	Top-down Design	Drawing a house
13	The Big Project	Specifications
14	Working on your project	Larger-size programs
15	Exhibiting the final project	Responsibilities
Freelesson 1	MAKER	Editing graphics
Freelesson 2	PAINTER and simple compiling	Creating scenes
Freelesson 3	Text screen organization	Formatting

FIGURE 2

LESSON NUMBER	TITLE NEW LANGUAGE FEATURES	PROJECT AND TEAM STRUCTURE
1	VAR parameters	Teams of 3, main program provided
2	WHILE..DO and CASE	NASASARK - writing to specifications
3	TURTLE GEOMETRY	Spiral drawing - team allocates sub-tasks
4	READ, READLN, USES	More figures, introduction to compiling
5	Statement delimiters	Predator-Prey simulation; team writes all parts; debugging off-line
6	User-defined functions, REAL number variables	Vinyl Tablecovers - teams of five; emphasis on team performance
7	Strings	Encryption - teams evaluate different break-downs of the problem
8	Scalars, CHAR, sets	Continue lesson 7 - completion of task through team's effort as a unit
9	File I/O	Inventory - redefinition of a problem for coding; more teamwork skills
10	Records	Continue Inventory project - building "user-friendly" programs
11	Global declarations	Mid-term project - local problem defined for teams (5) to program for customers
12	Two-dimensional arrays	Continue mid-term project - teams swap completed programs for testing/debugging
13	Recursion	Boxes Within Boxes - sample program done iteratively, then recursively
14	Basic	Bar Graphs - general program to convert raw data into scaled & labeled graphs
15	More Basic	Continue bar graphs - importance of output display as part of the design
16:		Final Project - students as a class write an Adventure game (or alternatively, a mailing list management program) to practice learned skills on a large project
17:	Segmenting	
18:		

ALGORITHMICS IN HIGH SCHOOLS

Jozef Gruska

Computer Research Centre
Dúbravská 3
842 21 Bratislava, Czechoslovakia

ABSTRACT

A new approach to computer science education in high schools is described. This approach is based on the following ideas. /i/ Education in computer science should have broad goals and should concentrate upon the core of the field, i.e. on the laws of programming. It should emphasize very systematic collection and analysis of ideas, notions, facts, and methods of lasting values and of the basic importance for general education. /ii/ The education should be based on similar ideas as education in mathematics, physics and chemistry. /iii/ What to teach and how to teach should be influenced more by the theory development than by the present-day hardware and software and by urgent practical demands. /iv/ A systematic approach to computer science education should be emphasized. /v/ Computers should play an important role in this education, similar to the role laboratories and experiments play in physics and chemistry education.

A textbook based on these ideas has been written and used since 1978. The goals and contents of this textbook are presented. A systematic approach to the synthesis of correct, elegant and efficient algorithms and programs are emphasized in this textbook. Experiences with this approach /tested since 1978/ is discussed.

1. INTRODUCTION

1. The underlying philosophy of this new approach is actually expressed in the title of this paper where the word algorithmics is used to mean computer science and to make clear that, as far as high school education is concerned, algorithms /and data structures and programs/ and not computers are the essence of the field.

We understand algorithmics as dealing with problems connected with synthesis, analysis, and realization of algorithms /on hardware using software/.

2. Several years ago a decision was made to introduce special optional programs into nonvocational high schools - they are called gymnasia - in Czechoslovakia and computer science was one of them. Each of these special programs is taught for 4 hours a week in the eleventh grade and for 6 hours in the last /the twelfth/ grade.

At the end of 1976 I was asked to make a proposal for the curriculum. The proposal was written and the following courses were established.

Grade 11	algorithms /2 hours a week/ programming /2 hours a week/
Grade 12	algorithms /2 hours a week/ programming /2 hours a week/ computer systems /2 hours a week/

The courses called "algorithms" and "programming" are actually only two parts of one course. The separation was made to indicate that at least half of the time is needed for practicing.

The curriculum was accepted by the Ministry of Education and in June 1978 the first official experimental textbook for the basic course was finished [1] and used since September 1978 in several schools. In the following years more schools started to follow the textbook. In June 1981 a new version of the textbook was published [2] in which some methodological changes were made taking into consideration experience obtained with the first textbook during the first two years. Also, in 1981 a textbook with more examples for programming for grade 11 was published [3]. The textbook for the course "Computer systems" for grade 12 is in print [4].

3. If one looks more closely at this "new approach" he/she certainly finds out that, from the computer science point of view, there is almost nothing new in this new approach. What is new, as far as we know, is that several ideas, notions, and methods which most people in the field consider very theoretical, which some people co

sider appropriate for the graduate study, which few people consider as just good for the undergraduate study, are considered here to be the ideas, notions, and methods on which a high school education should be based. A textbook incorporating these ideas has actually been written and used.

2. MAIN IDEAS

The main idea behind our approach to computer science education in high schools can be summarized as follows.

1. Education in algorithmics should concentrate on systematic collection and analysis of notions, ideas, methods, facts and habits of basic importance for general education /and by coincidence, also of crucial importance for computing/.

These notions, ideas, methods, facts, and habits should help students to develop:

a/ The art of systematically developing precise descriptions of activities needed to achieve desired goals with given resources /including the verification of correctness and the analysis of efficiency of the proposed solution/.

b/ The art and discipline of using clear, concise, and well-structured means in all communications.

c/ The art of systematically developing formal and clear descriptions of objects and systems and the art of preserving these descriptions in a form suitable for subsequent uses.

d/ The art of searching to find information necessary to solve given problems.

e/ Experiences in communicating with computers /input and output devices/.

f/ Habits in using computers to solve problems.

This main idea is based on the following observations and assumptions.

/I/ The arts mentioned in points /a/ - /d/ are necessary in any nontrivial job in science, industry, business, medicine, and so on. The degree a person comprehends these arts characterizes to a significant extent his/her ability to solve problems, to make plans, decisions, to deal with people, and so on. /II/ In the near future while most of the people will use computers, only few will really do programming.

It is quite clear which part of computer science education should help to develop these arts if properly taught.

/A/ Teaching a systematic approach to the synthesis and analysis of algorithms and programs should help to develop a systematic approach to problem solving and the art mentioned in point /a/.

/B/ Emphasizing clear, elegant, and well-structured descriptions of programs and their outputs should help to develop the art mentioned in point /b/.

/C/ A proper education in the area of data structures, data types, and data bases should help to develop the art mentioned in point /c/.

/D/ Sorting and searching /see point /d//.

Now we present the other ideas on which this approach to computer science education in high schools is based. The second idea is actually a consequence of the first one.

2. The basic education in algorithmics should concentrate primarily on a systematic design of elegant, correct, and efficient algorithms and programs which produce results in an easily readable form.

3. The basic education in algorithmics should be, as much as possible, independent of present-day software and hardware. For educational purposes a /special/ simple high-level programming language should be used which supports the initial intuitive process in the algorithm development, which allows to teach students "good design practices in a setting which is as simple as possible" [5], and which forces students to express programs and the otuputs of computations in a clear and elegant form. Computers /software and hardware/ should play a very important role in this education, similar to the role laboratories and experiments play in physics and chemistry education.

4. The basic education in algorithmics should be based on ideas similar to education in mathematics, physics and chemistry. The main part of the education should concentrate on systematic development and analysis of basic notions, ideas, facts, and methods in their simplest possible settings. Mathematics, physics, chemistry and other well established fields have learned that the way to achieve efficiency in understanding is to teach very basic but "toy concepts" - number, function, permutation, electron, gravitation, absolute motion, ideal gas, crystal lattice - and their application to small, idealized but "toy problems" with neat solutions, and not the construction and maintenance of big production plants. Computers and programming should play an important role in this education, similar to the role laboratories and experiments play in physics and chemistry education. Computers and programming should not be the main goal but the main support of this education.

5. What to teach and how to teach it should fall under the influence of present-day /but relevant/ computer science theory - as in mathematics, physics, chemistry - rather than under the influence of present-day /and not as relevant/ computer industry, especially software industry, and various important applications. because they pull computer science education towards methodologies and various very specialized computing techniques.

6. The basic education in algorithms and programming should make use of the education in mathematics and should support this education. The basic education in computer systems should make use of the education in physics /and chemistry/ and should support it.

7. A systematic approach to the basic education in algorithmics should be emphasized. Solving exciting and interesting

examples should play an important but supporting role. Developing large "real-life" software and endless playing with computers is for fans and "special interest clubs" similar to, for example, science fiction clubs. /Not only mathematics, physics and chemistry but also history, geography, biology and other fields have learned that the systematic approach to the education should be emphasized in comparison to reading exciting historical stories, playing war games, watching geographical movies, making exciting summer trips, keeping crocodiles, playing with monkeys and lions and so on./

The ideas of this section are very much along the ideas presented in [5] and [6].

3. METHODOLOGY

The ideas presented in Section 2 are quite general. In order to see more clearly what they mean and what should come out of these ideas, a concise description of the contents of the basic textbook based on these ideas is given in the next section. In this section we summarize the basic methodology which is used in that textbook and which, we believe, should be used in such an education.

1. An interative approach is used. Basic ideas, notions, methods, and problems are implicitly introduced in the very beginning. Gradually they are discussed again and again in a more precise, more complete, and more formal way. At the end they are used very explicitly in the process of constructing sorting programs.
2. All algorithms are systematically designed. None is just given.
3. Termination, correctness, and efficiency of algorithms and programs is discussed whenever possible and appropriate, as an integral part of their design.
4. A systematic approach to algorithmic problem decomposition is presented at the very beginning and is consequently applied.
5. A simple and very logical programming language is used which supports systematic program development and design of elegant programs.
6. Programming is presented as a four stage process: /i/ top-down decomposition of algorithmic problems, /ii/ choice of data structures, /iii/ optimization, and /iv/ testing.
7. "Abstract data type" approach to data structures is systematically used. Mainly algebraic arguments are used to introduce and discuss data types.
8. Flow diagrams are used only at the very beginning and only to visualize algorithms, never to construct algorithms.
9. A low-level machine language, FIP, is introduced but synthesis and analysis of FIP programs is done only using a high-level language.
10. Computer independent arguments are used primarily to introduce new notions and methods.

4. BASIC TEXTBOOK

In this section we describe goals and give the concise contents of all seven parts of the textbook.

4.1 ALGORITHM

Goal To introduce basic intuitive notions of an algorithmic problem /given as {input condition} ? {output condition}/ and of an algorithm as a solution of an algorithmic problem. To show how to read, to understand, and to execute algorithms. To indicate the structure of algorithms and to introduce basic techniques of a systematic approach to algorithm design.

Concise Contents

4.1.1 Greatest common divisor /GCD/. The algorithmic problem. Disadvantages of a table solution and necessity of an algorithm to compute GCD. Systematic construction of two algorithms. An implicit introduction of compound, if and while statements. Invariants, correctness, efficiency, and realization of two given algorithms.

4.1.2 Examples. Day of the year, the labyrinth, area calculation. Systematic design of algorithms. Representation of algorithms by flow-diagrams.

4.1.3 Description and execution of algorithms. An assignment statement and its components. Tracing. Decomposition of statements and blocks of flow diagrams by sequencing, branching, and cycling. The basic loop statement,

$$\downarrow\underline{\mathrm{repeat}}\ P_1\ \underline{\mathrm{while}}\ b \underset{\downarrow\,-}{\xrightarrow[+]{}}\ \underline{\mathrm{do}}\ P_2\ \underline{\mathrm{od}}\ \lrcorner$$

Invariants. Hoare semantics for assignment statements.

4.1.4 Construction of algorithms. The decomposition of algorithmic problems by sequencing and branching. Some hints on how to solve an algorithmic problem $\{Q\}$? $\{b \wedge I\}$ using a while statement with I as an invariant and b as a termination condition. An example - a systematic construction of an algorithm for the game NIM.

Comments A very informal treatment. An implicit use of the programming language PETRA.

4.2 COMPUTER

Goal To introduce the very basic ideas concerning computers in a simplified form. To make clear that programs have to be written in a precise, restricted and finally in a very detailed form. To obtain basic experience with executing programs on computers /with inputs and outputs/.

Concise Contents

4.2.1 FIP computer as a fictive random access machine with memory as an infinite two dimensional array with rows labeled by identifiers and columns by positive integers.

4.2.2 FIP language. A simple assembly language with instructions of the form DIV A;4:STO B[8]; JCZ 5. Macroinstructions for input and output of arrays.

4.2.3 The rules for a hand translation of PETRA statements to FIP programs. Examples: Fibonacci numbers, recognition of well-parenthesized strings.

4.2.4 An interpreter for FIP programs

4.2.5 The project. To design an interpreter for a subset of the FIP language in the FIP language. /For very good students only./

Comments All FIP programs are constructed first as algorithms in the PETRA language and then the translation is made. The FIP interpreter produces easily understandable listings with information on how many times various instructions were performed.

4.3 ANALYSIS OF ALGORITHMS

Goal To make clear that a program can run forever, can be incorrect, and not be the fastest one. To indicate methods to show termination, verify correctness, and to analyze efficiency. To help understand and deal with problems encountered during the execution of FIP programs on computers.

Concise contents

4.3.1 Termination Examples, methods to show termination.

4.3.2 Correctness An example of an incorrect program. Limits of testing. Step-wise construction of correct programs. An example - linear search. Correctness and testing of FIP programs - correctness of PETRA algorithms and correctness of their translation.

4.3.3 Efficiency. Tables showing computation time for algorithms of various time complexity. Time analysis of simple FIP programs /polynomial evaluation, x^m, binary search/.

Comments. This subject is dealt with while students are still busy with running FIP programs. No new ideas as far as algorithm construction in concerned are introduced here, only new questions are asked and answered. Time complexity of FIP programs is determined by analyzing PETRA algorithms.

4.4 PROGRAMMING LANGUAGE

Goal To describe programming languages in general and the language PETRA in particular. To obtain basic experience with running high-level language programs on computers.

Concise contents

4.4.1 The main characteristics of /good/ programming languages

4.4.2 Programs and their components. Constants, variables, expressions, statements, declarations, programs.

4.4.3 Syntax /syntactical diagrams/ and semantics /Hoare semantics for PETRA/.

4.4.4 Compiler /for PETRA/

Comments PETRA has been used implicitly from the very beginning. In this chapter PETRA language is described in a systematic and more precise way.

4.5 PROGRAMMING

Goal To describe in a more explicit and more clear way top-down decomposition of algorithmic problems. To present basic optimization techniques. To show how to organize computation outputs /tables, graphs/.

Concise contents

4.5.1 Top-down decomposition of algorithmic problems /sequencing, branching, and cycling/. Case and loop statements for multiple branching and cycling. Examples /transformation of numbers from a b_1-ary number system to a b_2-ary number system/.

4.5.2 Optimization of expressions, assignment statements, compound statements, branching statements, and cycles. The importance of internal loops optimization. go to statement and its use in optimization.

4.5.3 Output-organization Formats, output control commands.

4.6 DATA TYPES

Goals To introduce the most basic data types as sets and operations.

Concise contents

4.6.1 The notion of a data type and the meaning of declarations.

4.6.2 Elementary data types /integer, real, boolean, char/.

4.6.3 Data types array of D, array [M:N] of D, list of D where D is any data type. /List operations: first, last, next, previous, insert, delete, just?, last?/. For statement.

4.6.4 Examples. Search for the maximal element in an array, pivotisation.

Comments Quite a unified approach is used to describe elementary and structural data types. No artificial restrictions on the use of operations are made. Algebraic ideas are used more to motivate and to describe data types than memory considerations.

4.7 SORTING

Goal To show how ideas and techniques developed in the first six chapters can be used to construct systematically correct and efficient sorting algorithms and programs.

Concise content

4.7.1 Step by step sorting

First invariant: after the i-th step the last i elements are sorted.

Insert sort and binary sort.

Second invariant: after the i-th step the last i elements are definitely sorted.

Bubble sort and exchange sort.

Third invariant: after the i-th step the last i elements are definitely sorted and the first n-i elements form a heap.

Heap sort.

4.7.2 Divide and conquer

1. Maximally simple decomposition - merge-sort
2. Maximally simple composition - quick-sort

Comments All algorithms are first constructed working with the mathematical notion of a sequence. The choice of a data structure - an array or a list - to represent a sequence is then discussed, the rogram is written, optimized and analyzed.

The book contains about 800 exercises. It has 199 pages, 139 pages for the basic ext and examples, 60 pages for exercises.

5. EXPERIENCE

We will try to summarize the first experience with the approach presented in this paper.

1. The textbook appeared at a time when our high school education was concentrated on teaching Fortran or Basic by enthusiastic programmers. Most of the teachers were quite shocked by the new approach which was so different from the one they were used to.

2. The main issue being discussed was why not to use flow diagrams and not to use Fortran or Basic.

3. Most of the high school teachers received no education or insufficient education in computer science during their university study. We knew that and therefore the textbook is self-contained. Quite a few summer schools and special seminars were organized to help teachers, especially newcommers in the area, to understand what to teach and how to teach. Some of the main problems in teacher re-education were of an administrative nature.

4. A computer independent approach was generally welcomed by the teacher with no experience in the area but with experience in teaching mathematics. On the other hand they lacked the experience necessary to run programs on computers and were afraid of computers.

5. Gradually the approach has been generally accepted and mainly technicalities are discussed now.

6. The main suggestions: /i/ to avoid flow diagrams completely; /ii/ even more emphasis on a systematic approach to program construction and to force even more detailed documentation /in the form of invariants/ in the process of program development; /iii/ to introduce procedures sooner.

7. Some experienced teachers consider some parts - the second and the last chapter - too easy and they want more difficult examples for very enthusiastic students.

6. CONCLUDING REMARKS

1. In this paper we have discussed the underlying philosophy of our approach in general and the basic textbook in detail .

One should keep in mind that this is only the first part of the story. Computer science education continues in the twelfth grade where the following subjects are covered.

Algorithms /and programming/: /i/ functions and procedures; /ii/ algorithms of numerical mathematics; /iii/ more on data types /stacks, trees, files/; algorithms of data bases.

Computer systems: /i/ introduction /basic notions, multilevel, computers, hardware and software, processes and processors/; /ii/ computer architecture /main parts, basic structures, CPU, information representation, operations, memory/; /iii/ conventional computers /examples - IBM 360/370, PDP 11, VAX 11 - data representation, addressing, instructions/; /iv/ microcomputers /target and host computers, microoperations, interpreter/; /v/ operating systems /goals, accesses, peripherals, resource allocation, language implementation/.

It is true that problem solving and design of algorithms, data structures and programs are emphasized in our approach. However, it does not mean that the importance of computers and computer literacy is ignored. The reverse is the case. Computers should be used very intensively from the very beginning. /The textbook does not mention them because there is such a variety of computers in high schools./ And not only that. There is quite a detailed treatment of computer systems in the twelfth grade /using already developed algorithmic background/. Our approach only puts design of algorithms in the first place and the study of computer systems in the second place.

3. Our approach requires that students design and run a lot of small programs written in a simple and easy-readable language. It means that this approach may be used at schools with very little computing equipment. On the other hand this approach is intended primarily for schools /and societies/ with very good computing facilities, especially with a lot of terminals or microcomputers or personal computers.

4. Our approach is based on the idea that education in computer science should have broad goals and should be oriented to problem solving methods and to the design of programs in a very broad sense. This is in accordance with the ideas [6] that "we live in a world of programming and keep programming most of the time, whether we realise it or not" and that "The world of programs is by no means confined to the staff fed into computer memories. It is above all the enormous stock of operational knowledge accumulated by mankind, which is being merely actualized now in computers, robots and automatic devices. A still greater amount of programs is stored in the gene pool of all the living creatures."

5. One of the criteria for a good basic education in computer science is that this education is not essentially influenced by new technological developments. If we look at our approach from this point of view with respect to the present-day revolution in microelectronics, then we see that new /VLSI/ technology has no essential influence on our basic approach. This technology causes that microcomputers are faster and cheaper and are more easily available for high-schools and students. On the other hand, the fact

that this new technology makes a revolution in computer science, [8], does not seem to have an effect on our basic approach. However, it may influence the subject "computer systems" where also VLSI or even systolic algorithms may be mentioned.

6. Our approach is in accordance with the present-day views on the role of programming for general education [6, 7] and on the basic concepts of computer science education [5, 9]. On the other hand, our approach seems to be quite different from ACM recommendations/for a one-year course/ [10], which emphasize more computers than programming and if programming, then in Basic.

7. REFERENCES

1. J. Gruska, M. Franek: Algorithms, /In Slovak/. SPN Bratislava, 1979
2. J. Gruska: Algorithms, /In Slovak/, SPN Bratislava, 1981
3. M. Franek: Programming, /In Slovak/, SPN Bratislava, 1981
4. R. Filustek, S. Síma: Computer Systems. /In Slovak/, SPN Bratislava, 1982 /in print./
5. J. Nievergelt: Computer science education: an emerging consensus on basic concepts. Information Processing 80, S.H. Lawington /ed./, North-Holland 1980, p. 927-933
6. A. P. Ershov, G. A. Zvenigorodskij, J. A. Pervin: Informatics in schools /In Russian/. Report, Computer Centre in Novosibirsk, 1979
7. A. P. Ershov: Programming, the second literacy. In "Computers in education", Proceedings of 3rd World Conference on Computer Education, North Holland 1981, p. 1-7
8. 1. E. Sutherland, C. A. Mead: Microelectronics and computer science, Scientific American, 1977, p. 210-228
9. D. Gries: Educating the programmer: notation, proofs and development of programs. Information Processing 80, S.H. Lawington /ed./, North-Holland 1981, p. 935-944

10.J. B. Rogers, R. H. Austing: Computer science in secondary schools. In "Computers in education", Proceedings of 3rd World Conference on Computer Education, North-Holland 1981, p. 651-656

AN ORIGINAL HIGH SCHOOL PROGRAMMING COURSE THAT WORKS

Bill Shirley
7909 E. 67th Street
Kansas City, MO 64133
(816) 356-0657

"New occasions teach new duties" was a statement made by James Lowell a century ago, and when planning a semester course in computer programming, we certainly reaffirmed those words. Our goals were simply stated: to teach students to write programs in Basic; to help them discover whether they had an interest and or an aptitude in programming; and to show them that neither experience need be unpleasant.

The first hurdle was to find a text compatible with our nine Apple computers and our preconceived notions on the general flow of the course.

Either we gave up too soon or no such text existed, so we made our first decision We wanted to find a book to supplement the day-to-day discussions and simply write the course to suit our own priorities and style.

The book we chose was Basic From The Ground Up by Donald Spencer. It seemed to have good discussions and reasonable examples in those areas we thought would need reinforcing, such as sorting and alphabetizing, subscripted variables, etc.

We compiled a list of all the instructions and commands we would introduce, then hammered out the order of presentation that seemed most logical to us.

The general outline of a typical day was to have a student put the previous day's assignment on the chalkboard or overhead and discuss it if necessary. The new material would then be presented, and a handout with a sample program relating to the new material would be given to the student. They would then go to the computers in groups of three to type in and execute the sample.

They were constantly urged to change and modify the sample, discuss it within their group, and share any significant observations with the entire class. This last point, along with keeping a notebook of these observations and the daily discussions, was to prove to be a true time saver, for many of the items that were scheduled for future discussions would be disposed of, and we could accelerate our timetable.

Once the students had milked the sample dry, they would return to their seats and receive a second handout consisting of an assignment related to the sample.

Our governing principle throughout was to keep the lecturing to a minimum and allow the "hands-on" activities to dominate the 55 minute period.

As it was, the best mix was to approximate a 15-40 split. That is, 15 minutes of teacher-student interaction and 40 minutes of time spent at the computers. In the 40 minute portion, the teacher would be free to assist those students most in need of guidance. Several examples of the daily activities follow.

Day 1

OBJECTIVES:

1) To give students an initial sense of the parameters of a computer.

2) To give students the opportunity to interact within a group.

3) To give students immediate "hands-on" experience at the computer.

4) To experience typing and running a program.

5) To construct deferred instruction statements using "PRINT","X=", "IF THEN", "GOTO","END".

6) To use immediate commands LOAD,LIST, RUN, HOME, NEW, SPEED.

7) To learn simple modifications on a computer program.

ACTIVITIES:

1) Read "What a Computer Doesn't Have-Brains," (pages 1-5) in Basic From The Ground Up.

2) Assign students in groups of three

to a computer.
3) Load in the game "LEMONADE" and run for 10-15 minutes.
4) Type and run Example 1.1 (Refers to assignment, first day, first problem).
5) Discussions of Example 1.1.
6) Hand out Assignment 1.1 (Refers to assignment, first day, first assignment).

EXAMPLE 1.1:

Type in EXACTLY as printed. Strike the "RETURN KEY" as designated by (RTN).

```
10 X=1  (RTN)
20 PRINT X  (RTN)
30 X=X+1  (RTN)
40 IF X=11 THEN END  (RTN)
50 GOTO 20  (RTN)
RUN (RTN)
```

Day 3

OBJECTIVES:
1) To learn the use of commas and semicolons in print statements.
2) To observe use of the blank print statements.
3) To read and analyze a program.
4) To observe the difficulties arising from the computer's use of LOGARITHMS.

ACTIVITIES:
1) Execute Assignment 2.2 in groups to check program.
2) Discuss and check Assignment 2.2 and 2.3.
3) Execute and discuss Example 3.4.
4) Discuss difficulties of log calculations.
5) Hand out Assignments 3.4 and 3.5.

EXAMPLE 3.4:

```
10 PRINT "#","SQUARE","CUBE"
20 X=1
30 PRINT X, X^2, X^3
40 PRINT
50 X=X+1
60 IF X=11 THEN END
70 GOTO 30
```

ASSIGNMENT 3.4:

Assign 1 to K1, 2 to K2, 3 to K3, 4 to K4. Write a program which would produce each of the following output.

```
A.  1 B.  1 2 3  C.  1 2  D.  1    E.   123
    2                3 4        2
    3                              3
```

ASSIGNMENT 3.5:

For each program segment, figure out the final value of each variable. Do not execute these programs on the computer.

```
1.  10 Z=3      3.  10 A3=4
    20 Y=3+7        20 A4=3
    30 Z=8          30 A4=A3
    40 X=Z          40 A3=A4

2.  10 M=3      4.  10 X=10
    20 Q=M+M        20 Y=4+8-X
    30 M=Q          30 Z=Y+3
    40 Q=M+M        40 X=3*(X 2+5)
                    50 Z=Z-S
```

Day 7

OBJECTIVES:
1) To review major concepts and commands presented in Days 1 through 6.

ACTIVITIES:
1) Restate review list and answer any questions.
2) Hand out and explain Project Sheet.

PROJECT SHEET

Five projects from this list are due by the last week of the quarter.

All projects are weighted the same.

The five projects will comprise 40% of your grade.

More than five projects will count as bonus points.

BE CAREFUL! Exchange of ideas is encouraged, but plagiarism will be dealt with forcefully.

---

1) Input positive numbers X and Y. Compute the product of X and Y using only addition.

2) Input 20 numbers. Sum the negatives and sum the positives separately. Print out the two sums and tell how many of each kind of numbers there were.

3) A counting number X is a perfect square if for some counting number Y, Y*Y=X. Input a counting number X and decide if X is a perfect square.

4) You have 1000 feet of fencing and want to create a rectangular pasture. Using all the wire, determine the length and width for maximum area. Use some form of "brute force."

5) Write a program which inputs two integers, decides which is the smaller, then outputs a table of squares and cubes of the integers from the smallest to the largest.

6) Input 2 positive integers X and Y. Output the value of the Yth digit of X (from the right). Note there are some problems. i.e. Does X have at least Y digits? How large can Y be?

7) Input 3 integers and determine if they could be the lengths of the sides of a triangle. Hint: In Geometry you learned, in a triangle, the sum of the lengths of two sides must be greater than the length of the third side. Your output should indicate the 3 lengths and tell if they do or do not constitute the lengths of the sides of a triangle.

8) Input the price of an item in cents (< =99), output the number of quarters, nickels, dimes, and pennies a person would receive in change if he paid for the item with a dollar bill.

9) Design and write a program that will print a chart that will tell yearly, monthly, weekly, daily, and hourly pay rates based on a 40 hour week, 1/12 year month, and a 52 week year. Do this for yearly incomes from $16,000 to $25,000 in

$1,000 increments. You may use TAB to have the chart have all 5 columns side by side and appropriately labeled.

Day 10

OBJECTIVES:

1) To learn the process of rounding off.
2) To use the spiral technique to refine past concepts.

ACTIVITIES:

1) Hand out Example 10.9.
2) Hand out Assignment 10.8.

EXAMPLE 10.9:

```
10 X=3.14159
20 X=100*X
30 X=INT(X+.5)/100
40 ?X
50 END
```

ASSIGNMENT 10.8:

Write two programs, one to modify Example 3.4 and one for Example 4.5(f). Each had a "snag" which rounding off might cure.

Day 26

OBJECTIVES:

1) To learn the READ-DATA commands.
2) To re-emphasize variables, TAB and style of output.

ACTIVITIES:

1) Hand out Example 26.20. Discuss.
2) Hand out Assignment 26.18.

EXAMPLE 26.20:

```
   10 ?"ITEMS";  TAB(8) "UNIT PRICE";  TAB
(20) "COST";  TAB(27) "SUBTOTAL".
   20 DATA 2,.35,3,2.65,11,.25,1,9.49, 35,
1.59
   30 READ I,U
   40 T=T+I*U
   50 ?TAB(2)N;TAB(12)U;TAB(20)I*U;TAB(28)T
   60 GOTO 30
   70 END
```

ASSIGNMENT 26.18:

Modify Example 26.2 to total the number of items and label as follows:

There are ___ total items.

The truly mystifying result came when we had introduced all the instructions on our list and found it had consumed half the semester almost to the day.

We therefore determined that the 1st quarter grade would be based on the results of 3 tests interspersed on days 8, 21, and 45 for 60% of the grade. The 5 projects mentioned on day 7 would account for the remaining 40%.

Before and during this time, we were most concerned whether this pace was too fast. But the students assimilated the material and recalled items at an incredible rate!

In fact the tests gave a true indication of this. Each test was divided into three parts. One was to evaluate the syntax and basic definitions with short answer, true-false, and mulitple choice questions. The second portion of the test was to analyze a given program and predict the output. The third involves the student writing a complete program satisfying given data.

The failure rate was quite low. As of this writing, 9% of those students who enrolled in the course failed 1st quarter. The ability difference in students was evident mainly in the amount of assistance needed in analyzing the initial steps in writing the 5 projects. However, since this was a minority of the students, it caused no major problems.

It was then we initiated the portion of the course which excited us the most.

There were two goals for 2nd quarter: to review over and over the instructions introduced in the 1st quarter and to allow the students to experience writing programs from diverse areas. Holding to Ralph Waldo Emerson's thought that nothing great was ever achieved without enthusiasm, we plunged into a very unstructured setting.

The 2nd quarter would consist of strictly one-on-one interaction between the student and the teacher. Initially, the student would select a project from a furnished list. If students needed assistance, they would take a number, much in the fashion of a catalog order desk at Sears. When the teacher finished with the student, the next number would be called out.

When the students felt the program was arriving at a workable stage and after a cursory inspection by the teacher, they would type in and execute their program on the computer. After making a hard copy on the printer, the students would return to their seats and concentrate on de-bugging, cleaning up the output, etc. This proved to be the most cantankerous problem of the course--getting the student to write the programs away from the computers. It was truly a struggle to convince them of having a hard copy before they sat down at the station.

Following is the synopsis page for the 2nd quarter activities along with a sampling of the projects from the list.

SECOND QUARTER

The second half of the semester will be devoted entirely to the writing of programs.

The projects will be separated into two levels.

The projects in levels are numbered 1 through 7 indicating difficulty and point value. i.e. 1 is 10 points and less difficult, 2 is 20 points and somewhat harder- 3 is 30 points, etc.

		No. points	Total
Level 1	1	2 @ 10	20
	2	5 @ 20	100
	3	9 @ 30	270
	4	10 @ 40	400
Level II	5	6 @ 50	300
	6	13 @ 60	780
	7	1 @ 70	70

50 problems; 1940 points possible.

Students are encouraged to initiate projects on their own, discuss them with the teacher, and arrive together at a point value. Grade Requirements:

495)
421) E At least 250 points from Level II

420)
346) S At least 200 points from Level II

345)
271) M At least 140 points from Level II

270)
196) I Problems selected from any Level

195) Failing

COMPUTER PROJECTS SECOND QUARTER

(Number in parenthesis indicates difficulty level).

(1) 1. Input 10 numbers. Count the number of zeros that appear and print this total. Example; 4 zeros occur in the group 1;50;3;200.

(2) 3. Determine and print <u>all</u> the three-digit numbers which equal the sum of the cubes of their digits.

(3) 9. Given two positive integers, compute their greatest common factor.

(4) 21. String tabulation. (What the character is and how many times it occurs). Example: Input "What's happnin' baby?"

```
   Output:    Character      Occurs

                  A          3 times
                  B          2 times
```

etc.

(5) 28. Write a program to change any 3-digit number to base 2.

(6) 37. Input a positive integer and output its Roman Numeral representation.

The preceding was a sampling from the 50 projects.

There was, in spite of our enthusiasm for the routine just outlined, a certain amount of trepidation as to how the students would react to such an unstructured plan. What we learned was simply a verification of the old adage; students tend to seek the level of the teacher's expectation.

Less than 10% of the students failed to meet the minimum number of programs.

The student often felt it a badge of honor to be the first to attempt a given project and many vied for the highest point total. Enough excitement was evidently carried home that the parents asked for and received their own mini open house in order to visit the computer room with their child.

The sessions were well attended and quite lively. The student and their parents sat at a computer and reversed roles: the child explained the process and the parents asked the questions. We received nothing but positive feedback from the participants. At the request of the parents, the District is also planning to offer the course in the evening adult education program.

Both students and teachers felt the course was an enormous success! In a critique of the course, the students expressed a sense of knowing the parameters of the computer. Over 12% were considering Computer Science as a program of study in college. Even those unable to generate interest in the field felt the course had been worth their time and effort.

The National Council of Teachers of Mathematics, for several years, has stressed the need for more "problem solving" to be taught in secondary schools. This course seems to respond to that call very well, the gradual awareness of the student for the need of careful analysis of given data and the concantenizing of the instructions into a coherent program all lead the student to a greater understanding of the art of problem solving.

Special Session

EDUCATORS FOR COMPUTING AND HEALTH CARE A NEW PROFESSION

Sponsored by: ADCIS
Chaired by: Xaren A. Duncan
Health Information Systems

ABSTRACTS

Computing Applications have been developed in virtually every area of health care. But technology transfer is lagging because of shortages of skilled health computing teachers and other personnel. This session presents an overview of career opportunities in the field, with special emphasis on the need for more educators. The recently published Report of the ACM Committee on Curriculum for Health Computing Education on Guidelines for Doctoral-Level Programs in Health Computing will be discussed, with special emphasis on teacher training, computer-based learning, and administration.

Participants

Sidney Katz
Medical University of South Carolina

Richard E. Pogue
Medical College of Georgia

Special Session

ORGANIZATIONAL SUPPORT FOR PROFESSIONALS INTERESTED IN PRECOLLEGE EDUCATION

Sponsored by: ICCE
Chaired by: Dick Ricketts
The Computing Teacher

ABSTRACT

Professional computer (and other) organizations are playing a vital role as our schools enter the computer age. Panelists will tell what their organizations are doing to assist educators and to further computer instruction. They also will discuss what new efforts may be necessary if our society is to make the most of computers.

Participants

James M. Adams, Jr.
Director of Operations, ACM

Mrs. Bobby Goodson
Pres. Computer Using Educators (Calf)

Dave Moursund
Pres. ICCE

Lary K. Smith
Pres. Mich. ACM
Communications Secretary, Mich.
Association for Computer Users in Learning

Dennis W. Spuck
Pres. AEDS

Project Presentation

PRECOLLEGE TEACHER EDUCATION

Chaired by: Doris K. Lidtke
Towson State University

Larry Dillingham
Philip East
Ted Mims
Tillman J. Ragan
Rosalle Rhoads
Nancy Roberts
Robert Underwood

ABSTRACT

A Computer Education Course for Elementary Teachers

Philip East, Dept. of Computer and Information Science, University of Oregon, Eugene, Oregon 97403

Teacher education programs are becoming increasingly aware of the need to include computer education. The near total lack of background on the part of pre- and in-service teachers, however, requires extensive remediation. Unfortunately, there is neither room in the teacher education curriculum to accomplish this task nor the knowledgeable faculty if room were available.

This situation is predicated on the assumption that teachers must fill the role of content expert in their dealings with students. A single course may well supply a short term solution to the computer education problem; if teachers need not play the role of content expert. The proposed course attempts to prepare teachers to be non-experts in computer education.

As non-expert in an unfamiliar role, teachers will need to be eased into it. Thus, the course provides a basic broad knowledge that will sometimes suffice in teacher-pupil interactions. For the times when a greater depth of knowledge is necessary, teachers will have been prepared to involve students in the process of finding the knowledge. There are, then, two major goals of the course.

First, the imparting of basic knowledge related to computers and computer education. This is accomplished in two ways. Some class lectures/discussions and readings will focus on knowledge. Topics include: what computers are and how they work; computer applications in society and related issues; and computer applications in education. In addition to reading about and discussing these topics, there will be considerable hands-on experience with a variety of educational applications.

The second goal centers on the process of learning. Throughout the course, participants will be learning without direct supervision. Discussion (before, during, and after) this process will explicitly examine what is involved in learning about and using computers. Other discussion will attempt to extrapolate the process to elementary students.

Such a course ought to provide an adequate short term solution for teacher education programs wanting to incorporate computer education. Its underlying philosophy, that education should teach learning, is, however, deemed worthy of consideration beyond the confines of this particular course.

ABSTRACT

Presentation of Topics and Programs for Elementary and Secondary School Administrators

Ted Mims, Department of Computer Science, Louisiana State Univ., Baton Rouge, LA 70803

The presentation consists of handouts, overhead transparencies, computer programs and lecture methods. The presentation is intended to help increase the computer literacy of school administrators.

Topics presented and discussed include:
What does a school administrator need to know to be computer literate?
How can computers improve public relations?
How much power does the data processing director have?
What type of personnel to hire to teach computer science?
What type of computer is needed for instructing?
What type of computer is needed for administrative purposes?

Typical programs available for school administration.
Characteristics of a good administrative program.
Demonstration of a data base management system.

ABSTRACT

An inductive Approach to Learning CAI Programming in BASIC

Larry M. Dillingham, M.Ed., Instructional Designer; Tillman J. Ragan, PhD, Prof., Educational Technology, University of Oklahoma, 820 Van Vleet Oval, Norman, OK 73019

"The Microcomputer in Teaching" (Educational Technology 5113) is a course in programming and the educational uses of microcomputers taught at the University of Oklahoma. The course approaches the learning of computer skills in an unusual and highly effective manner. This presentation will explore the inductive instructional strategy employed, showing the participants the techniques methods, and materials which are used to deliver this effective, aptitude-independent course.

Experience with the course indicates that students who typically have difficulty learning computer programming do not have difficulty learning the skills as they are presented in the course. The essentials of course delivery include:

a) learning from program models
b) intensive computer usage
c) directed inference
d) CAI development in BASIC

LEARNING FROM PROGRAM MODELS

A great deal of initial learning of programming is stimulated by use of model programs. A short CAI module (listing provided in the presentation) which makes use of short, single task subroutines allows the beginning programmer to produce a CAI program incorporating his or her own subject matter. After exploring this program, the course moves on to the WAY the program works, learning BASIC words and techniques as needed.

INTENSIVE COMPUTER USAGE

The course is conducted in a microcomputer teaching laboratory equipped with 9 Apple II computers with disk drives. Class size is limited to 18 students to allow for maximum computer time for each student during every class period. The class is taught in 3-hour blocks.

DIRECTED INFERENCE

Directed inference is a descriptor for the process by which students translate routines on a model into a concept of "what does what" in the computer programs. Rather than learning the meanings of the words in BASIC, students are taught blocks of code which have a specific function. This "functional" learning is later transitioned into learning to program in BASIC.

CAI DEVELOPMENT IN BASIC

The course is designed for teachers and education students with no computing experience. It therefore concentrates on the skills necessary for developing Computer-Aided Instruction. All programming is done in the BASIC language so that graduates of the course may go directly to an "off-the-shelf" microcomputer and apply their skills and knowledge. The fact that a student is able to write workable CAI programs after the first block (3 hours) of instruction leads the authors to describe the course as "aptitude independent".

ABSTRACT

Microcomputer Usage for Special Educators and School Counselor-Development of a Symposium

Rosalie Rhoads, PhD, Susan M. Zgliczynski, University of San Diego, Alcala Park, San Diego, California 92110

This symposium was just organized because of a realization that special educators and school counselors were often neglected in existing programs on computer literacy for educators in the San Diego area. At our own university we were beginning to integrate a computer literacy module in our preservice teacher education program, but our graduate students in Special Education and School Counseling were not exposed to the use of microcomputers.

Inservice training on micro was being conducted by the local city and county schools, but applications specific to special education and counselors were not included.

Investigation revealed that in these two areas of student services, development of hardware and software had been supported but no one had trained teachers to use these new materials.

For years the U.S. Office of Special Education (OSE) has funded research and demonstration projects for the development of technology for improving the education of handicapped children. In 1981, OSE began a new grant program to investigate variables which foster the use of education technology, but this information has not filtered down to schools or universities in our area. So there had been developed in psychological testing, study skills, career planning, preparation for standardized tests. Few counselors in our area were familiar with these developments.

We scheduled a joint symposium for special educators and school counselors. An all-day Friday program provides hands-on training in setting up a microcomputer, using the keyboard, loading software, and writing a simple BASIC program. Saturday's program includes sessions on using career/life planning software, learning how testing and test preparation can be done on the computer, writing and updating IEP's on micros, demonstration of technological innovations that help handicapped students learn on the computer, and demonstration of software to develop study skills and remediate reading and writing deficiencies.

The project presentation for NECC will outline a preservice/inservice model of teaching special educators and counselors to use technology in the classroom.

ABSTRACT

A Master's and CAGS Degree Program: Computers in Education

Nancy Roberts, Graduate School of Education, Lesley College, 29 Everett Street, Cambridge, MA 02238

Lesley College, Graduate School of Education, is completing its second year offering a graduate program in Computers in Education. The thrust of the program is to provide precollege-level teachers, especially those without a science or mathematics background, the competence needed to assume a variety of computer-related roles upon graduation, including:

- the computer resource person for a school district
- the organizer of a high school computer science department
- a designer of computer courses for training employees of a private or public sector organization
- a teacher trainer
- an author of computer related books, manuals, and journal articles

The course offerings, which are continuously evolving, currently include:

Computer Literacy/Introduction to BASIC, Logo, DYNAMO includes an introduction to the history, design and logic of computers as well as an overview of three computer languages.

Computers in the Schools/Applied Uses for Teachers, Administrators and Specialists includes an introduction to data bases, files, record keeping and the issues involved with implementing management systems in schools.

Programming in Pascal/Intermediate Programming Techniques includes a hands-on experience at the computer terminal learning the Pascal language. Also included is a discussion of software development in general and the connection between problem-solving and computer programming.

Creating and Educational Environment with Logo introduces the student to the educational philosophy that has developed over the past 12 years at the MIT Logo Laboratory. Includes extensive exposure to the Logo interactive graphics system for programming, mathematics, animation, music, physics and text editing.

Computers and the Special Needs Student demonstrates the unique impact computers can have on the education of students with a wide spectrum of special needs.

Introduction to Computer Simulation focuses on the use of the computer as a laboratory tool to aid in understanding real-world phenomena. Students define problems for model building, conceptualize system structure, write model equations, and carry out simulation experiments.

Evaluation and Design of Educational Software focuses on the effective use of computer capabilities, standards for documentation, appropriate pedagogical strategies, programming aids, and evaluation in the process of developing educational software.

Fundamentals of Computer Structure includes the study of how computers work and how they relate to the world. Also included is the use of peripherals and an introduction to machine language.

Programming in BASIC includes development of skill in the use of the BASIC language and its use in educational applications to instruction and administration. Students are taught pseudocode as a tool for developing computer programs.

These courses will now be offered around the country through Lesley College's new National Outreach Program.

ABSTRACT

Overview of Two Mini-Courses and Two Conferences in Educational Applications of Microcomputers for Inservice Teachers

Robert Underhill, Kansas State University, 253 Bluemont Hall, Manhattan, KS 66506

The speaker has developed a one-semester hour course and a two-semester hour course. In both courses, in-service teachers have considerable hands-on experience with Apples, Commodores, and TRS-80s.

The focus of the one-semester hour course is awareness and rudimentary programming skills. In-service teachers learn to load and make back-up copies of programs. They also learn how to evaluate software, become acquainted with several categories of microcomputer use, become acquainted with microcomputer terminology, and write simple programs. A 60 page booklet has been written by the speaker for use in this course.

In the two-hour follow-up course, in-service teachers continue to explore various microcomputer uses in educational settings and to read about and discuss issues related to utilization such as criteria for purchase, and how to help colleagues become more aware. In this course, in-service teachers also extend their program-writing skills.

The Kansas State Department of Education has established six regional Microcomputer Users Groups. The speaker is the chair of the North Central Kansas group and will briefly discuss the format of two very successful conferences for public school teachers and administrators. One was held from 1:00 p.m. - 8:30 p.m. on a weekday, and one was held from 9:00 a.m.-4:40 p.m. on a Saturday.

The speaker will present details of the course contents, structure, and resources and will demonstrate some sample student programs. Also, information will be presented about the Microcomputer Users Group conferences.

Teacher Training

SUMMER INSTITUTES
IN MICROCOMPUTING
FOR
SECONDARY TEACHERS

John M. Baker
Western Carolina University
Cullowhee, NC 28723
(704)227-7385

INTRODUCTION

As the onslaught of microcomputers continues to spread, the cost of equipment fades as a prohibative factor, and pressure from parents and administrators builds to fill an apparent need, secondary teachers of mathematics are viewed as desperate in their need to quickly obtain the background, training, and support which will place them in position to act. The numerous publications pointing to this need and the most current recommendations for filling it [2] served as the springboard for two N.S.F. funded two-week institutes held in eastern and western North Carolina, during the summer 1981. It was administered by Charles Martin and taught by Charles Burrell, John Thomas (student assistant), and the author, all of Western Carolina University. Approximately 47 participants for each 4 semester hour institute were accepted from a total of 140 applications.

A description of the institutes follows with a report on follow-up workshops held in December 1981, and an analysis of the data collected from the participants which attempts to measure the efficacy of the project.

OBJECTIVES

In keeping with the suggestions made to us by local teachers and the literature at large, the decision was made to have a two-prong objective which was, in brief, to teach (1) Basic programming and (2) computer literacy. The particulars of the objectives were made to coincide with those of [2, p. 21, 66-70]:

B.1. Be able to write and document readable, well-structured individual programs and linked systems of two or more programs;

B.2. Be able to determine whether they have written a reasonably efficient and well-organized program;

B.3. Understand basic computer architectures;

B.4. Understand the range of computing topics that are suitable to be taught as well as the justification for teaching these topics;

B.5. Know what educational tools can be uniquely employed in computer science education.

B.6. Develop the ability to assist in the selection, acquisition, and use of computers, interactive terminals, and computer services which are suitable to the enhancement of instruction; and

B.7. Be able to assist teachers in evaluation, selection, or development of appropriate instructional materials which utilize computing facilities.

L.1. Understand classes of hardware, to include the major computer components.

L.2. Understand the relationship between algorithms and programs.

L.3. Interpret the role of software in data processing.

L.4. Identify applications and include language suitability, speed and cost considerations, and information systems.

L.5. Recognize the impact of computers on the individual and society.

L.6. Insure a positive attitude toward and motivation for learning more about computers and using computers.

It is accurate to say that virtually all of these particulars were treated during each institute.

EQUIPMENT/FACILITIES

Room A:	for Basic lectures and programming lab; 15 Apple 48K microcomputers with one drive each; 10 TRS-80 Model I 16K with one drive each; one 21-inch color monitor; one DATASOUTH DS-180 printer.
Room B:	for literacy lectures and demonstration of commercial software; one TRS-80 Model I 48K with two drives; four 19-inch black and white monitors.
Software:	Radio Shack Essential Mathematics (grades 8-12), Calculus, Euclid Geometry Tutor, Graphical Analysis of Experimental Data, Advanced Graphics, Quick Quiz, Minnesota Educational Computing Consortium (MECC) demonstration disk.

INSTITUTE FORMAT

Each of the two institutes was conducted under the format given below except for one change. Call it first day jitters, computer anxiety, or some form of "cyberphobia" [3, p. 46], it was found that at least five of the 47 participants in the first institute were suffering from extreme concern that too much material was being taught too fast in the lectures on Basic to be assimilated in the time allotted on the microcomputers. In the second institute, more time was allowed for all participants to gain a nodding acquaintance with the microcomputers before proceeding lockstep with the Basic lectures. Presumably, this conquered the problem.

Daily Schedule

Between 9:00 and 2:00, there were four lectures alternating between literacy and Basic, and the remaining afternoon saw three groups alternating between these activities: examination and critique of commercial C.A.I., description and assignment of programming homework, and free time on the microcomputers. In addition, the microcomputer room was open to the participants from 5:00 to 8:00 p.m. and 8:00 to 9:00 a.m. The schedule of lecture topics follows:

Day	Basic	Literacy
1	administration; variables, LET, PRINT	administration; growth in use of computers
2	REM, READ, DATA, GOTO, IF/THEN	mainframe, mini, micro classes; identification of segments of computer
3	flowcharts, INPUT counters	software: copyrights, warranties, liabilities; change in copyright law
4	FOR NEXT, TAB, intrinsic functions	information utilities; networks; antitrust actions
5	string variables, subscripted variables	international considerations; dwindling importance of IBM domination of mainframe market
6	subscripted variables, subroutines	survey of micro manufacturers; future trends in equipment
7	graphing, ASCII characters	where and why computers are used; factors limiting use--cost and human factors
8	PEEK, POKE	file structure; data banks; system security
9	study of a program listing handout	crime and the computer; information systems and privacy
10	wrap-up, administration	"dangers in the application of info mation technology t education" 1, p. 9 wrap-up

Remark

At the conclusion of each institute, we received a very solid impression that we had achieved our objectives. A great number of the participants had expressed the fact that they were attending the institute out of desperation. Many were already scheduled to begin teaching Basic in the fall having had no formal training and feeling very insecure. Others had received memorandums from administrators asking, out of the blue, for their plans for teaching computing in the future. From Day 1, an air of urgency had been felt. But after ten days of close association and tight schedules, was the message of achievement we were receiving a lasting one? In particular, is a two-week institute sufficient to provide the mathematics teacher with the background necessary to start a computer curriculum in the school?

FOLLOW-UP WORKSHOPS

In their applications, the participants had agreed to attend follow-up, one-day workshops in December. Letters of announcement were sent on November 1 calling for presentations at the workshops and submission of abstracts to be duplicated, bound, and distributed. This was not a good approach--we received only two abstracts. However, at the workshops a great deal of informal exchange did take place and several formal presentations were made. Participants had a scheduled time for sharing non-copyrighted software. In an attempt to measure the effectiveness of the institutes, the following questionnaire was administered, given here in facsimile form with the average response and tallies included for the 64 respondents:

1. Rate yourself as to how competent you felt, prior to the institute, as students, peers, and administrators looked to you for leadership and direction in educational computing in your school with respect to:

(a) Teaching Basic:
1 2 3 4 5 6 7 8 9 10 Average Response: 2.8
(b) Teaching Literacy:
1 2 3 4 5 6 7 8 9 10 Average Response: 2.8
(c) Being a Source of Information for Your Peers:
1 2 3 4 5 6 7 8 9 10 Average Response: 3.1
(d) Writing CAI for Your Own Classes:
1 2 3 4 5 6 7 8 9 10 Average Response: 2.3
(e) Having a Knowledge of Available Software:
1 2 3 4 5 6 7 8 9 10 Average Response: 2.5
(f) Having a Knowledge of Available Hardware:
1 2 3 4 5 6 7 8 9 10 Average Response: 3.1

2. Please address the same questions rating your self-confidence after the institute as you entered the new school year:

(a) Teaching Basic:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.4
(b) Teaching Literacy:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.7

(c) Being a Source of Information for Your Peers
1 2 3 4 5 6 7 8 9 10 Average Response: 6.9
(d) Writing CAI for Your Own Classes:
1 2 3 4 5 6 7 8 9 10 Average Response: 5.8
(e) Having a Knowledge of Available Software:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.6
(f) Having a Knowledge of Available Hardware:
1 2 3 4 5 6 7 8 9 10 Average Response: 7.2

3. Again, the same questions as it reflects your competency self-concept at this point in time.
(a) Teaching Basic:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.5
(b) Teaching Literacy:
1 2 3 4 5 6 7 8 9 10 Average Response: 8.6
(c) Being a Source of Information for Your Peers:
1 2 3 4 5 6 7 8 9 10 Average Response: 7.1
(d) Writing CAI for Your Own Classes:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.0
(e) Having a Knowledge of Available Software:
1 2 3 4 5 6 7 8 9 10 Average Response: 6.9
(f) Having a Knowledge of Available Hardware:
1 2 3 4 5 6 7 8 9 10 Average Response: 7.3

*Note: In 4. through 6., "NR" signifies "no response."

4. Before the institute, were you teaching a class
in Basic? YES 9 NO 51 NR 4
in Literacy? YES 7 NO 53 NR 4

5. After the institute, are you now teaching a class
in Basic? YES 22 NO 38 NR 4
in Literacy? YES 21 NO 39 NR 4

6. If you answered "NO" to either in #5, then go to the appropriate questions here--else go to #7

I plan, as best one can, to teach Basic
This spring 9 Some later year 20
School year 82-83 8 Never 3
NR 24

I plan, as best one can, to teach a course in computer literacy
This spring 6 Some later year 21
School year 82-83 7 Never 6
NR 24

7. What do you feel would be the most beneficial now to you:
2nd Institute 44
Newsletter 9
No Response 2
Individual consultation via phone/letter 0
University 3 semester hour computer science course 9

We could consider the institute a total success if, in each of the categories a to f of questions 1 to 3, we observed a strong increase in confidence between question 1 and 2 and moderate to no decrease between questions 2 and 3. Questions 4 through 6 were to help support or deny these observations.

DATA ANALYSIS

The data for questions 1-3 is summarized as follows:

Confidence Table

#	Average scores for question			change	
item	1	2	3	1 to 2	2 to 3
a	2.8	6.4	6.5	+3.6	+.1
b	2.8	6.7	6.8	+3.9	+.1
c	3.1	6.9	7.1	+3.8	+.2
d	2.3	5.8	6.0	+3.5	+.2
e	2.5	6.6	6.9	+4.1	+.3
f	3.1	7.2	7.3	+4.1	+.1

We do not wish to claim any profound significance in the above results, but we do offer some observations. Column 1 of the average scores reflects a very weak confidence level prior to the institute. Column 2 was not as strong as we hoped in measuring confidence immediately after the institute, reflecting increases in the change column which we term only "good." From conversations with the teachers during the follow-up workshops, we can describe the feelings reflected in the data as that of a good measure of increased knowledge and confidence tempered by a conservative caution. The real delight was in observing the third column of average scores and the corresponding change column. Where we had anticipated zero to small decreases, we saw small increases. This, along with conversations at the workshop, tends to say that when the participants reentered the teaching environment and exposed their knowledge by teaching new courses and interacting with peers, they were able to hold their ground. Turning to the data in questions 4 through 6, one observes that 22 inexperienced teachers are committed to teaching Basic and 20 to literacy during the 81-82 school year (approximately one-third of the participants). On the other hand, the "some later year" category of question 6 shows another third who have no definite idea on when they might begin teaching computing in their school. It is doubtful that lack of equipment is a controlling factor because applications for admission to the institute were screened for the presence or promise of presence of microcomputers. Moreover, applications were accompanied by letters of support from at least one administrator: Department Head, Principal, Superintendent. Thus, we express the opinion that for at least one-third of the participants, two weeks of intensive training is insufficient background for establishing a computer curriculum in the school.

If this is a valid thesis then it should be supported, to some degree, by a marked difference in confidence levels between the inexperienced teachers who committed themselves to teaching computing by no later than the 81-82 school year versus the inexperienced teachers who exhibited no commitment. Thus, we extracted the two subsets of the data defined specifically by

Inexperienced/committed:
4. Basic = NO and Literacy = NO, and
5. Basic = YES and Literacy = YES,
or 6. Basic = This spring and Literacy = This spring,
or 6. Basic = 82-83 and Literacy = 82-83

Inexperienced/uncommitted:
4. Basic = NO and Literacy = NO, and
6. Basic = later or never, or
Literacy = later or never

The confidence tables for these groups follows:

Confidence Table
Inexperienced/Committed

# item	Average scores for question 1	2	3	change 1 to 2	2 to 3
a	2.6	7.1	7.7	+4.5	+.6
b	2.3	7.1	7.7	+4.8	+.6
c	2.5	7.0	7.7	+4.5	+.7
d	1.9	6.4	6.8	+4.5	+.4
e	2.0	6.4	7.2	+4.4	+.8
f	2.7	7.4	7.8	+4.7	+.4

Confidence Table
Inexperienced/Uncommitted

# item	Average scores for question 1	2	3	change 1 to 2	2 to 3
a	1.8	5.9	5.9	+4.1	+.0
b	1.8	6.3	6.4	+4.5	+.1
c	2.3	6.7	6.9	+4.4	+.2
d	1.6	5.7	5.8	+4.1	+.1
e	2.1	6.6	7.0	+4.5	+.4
f	2.4	7.1	7.4	+4.7	+.3

The scores are noticeably lower for the second group. Although the first change columns (1 to 2) for both groups are comparable (saying that the individuals in each group felt they gained equally from the institute), comparison of the second change columns (2 to 3) reveals that the inexperienced/committed group felt they were more confident and had retained more, whence they were willing to commit themselves to act.

In summary, the institute participants were teachers who were so motivated to learn that they were willing to spend two demanding weeks of their summer in residence. Of these, approximately one-third gave indication of no commitment toward teaching computing after the institute. Waving aside the plethora of other lesser possible reasons, we cite insufficiency of the two-week institute.

REFERENCES

1. Licklider, J.C.R., Impact of Information technology on education in science and technology, Technology in science education--the next ten years, N.S.F., July, 1979.

2. Topics: Computer Education for Elementary and Secondary Schools, SIGCSE and SIGCUE joint issue, A.C.M. Education Board, January, 1981.

3. Wrege, Rachel, High (Tech) Anxiety, Popular Computing, Vol. I (no. 3), January, 1982.

QUALITY INSERVICE TRAINING

Steven R. Shotwell
Troy School District
4400 Livernois
Troy, Michigan 48098
(313) 689-0600

All the education journals said computers were the "new" answer in education. As a result, two Michigan school districts purchased microcomputers and their administrators gloated about how much they added to the curriculum. Not to be outdone, and also out of a sincere desire to be up-to-date, a neighboring district asked the school board to approve the purchase of microcomputers.

Now, each building in this district has five microcomputers, but they are usually kept in the media centers, off in a corner of the audio visual closet. Occasionally, a teacher will "sign one out" on a party day, and the classroom will reverberate with the blips and bleeps of Alien Invaders and Cosmic Commandos. The kids love it.

Once, when a teacher tried to use the computer as an adjunct to a math lesson, he discovered that he could not even find the ON/OFF switch on the keyboard. Other teachers mutter about how much these electronic monsters cost, even as the district is asking them to increase class sizes, denying new textbook purchases, or deleting other classes. "What good are these things?" they legitimately wonder.

An exaggerated scenario? Not necessarily. Sincere but short-sighted educators have made mistakes before. New Math? Language labs? Expensive AV equipment that only a few people can figure out how to use? If we are to be sure that computer education develops its potential and does not end up on the scrap heap of rejected notions, it is important that educators carefully plan inservice training and continually assess their needs for additional training.

As more and more school districts become aware of the exciting rewards that computers can offer to their curricula, they are faced with a common dilemma--quality inservice training. By definition, inservice training is the process of receiving instruction while on the job. Computer education, for the purpose of this presentation, includes all the various terms that educators have used to title their programs: Computer Assisted Instruction (CAI), Computer Managed Instruction (CMI), Computer Augmented Learning (CAL) or any other related title. The overall success of any new educational program in a school district rests on the quality of inservice training for those who are directly, and indirectly, associated with the program. Computer education is no exception to this rule. Administrators, teachers, and school board members must all become actively interested and involved to make computer education an ongoing success.

With this in mind, inservice training is necessary for all the groups involved, and should be tailored somewhat for each. For example, the teacher requires hands-on skills and implementation techniques, while the administrator needs to know the benefits, implementation methods, and management techniques for a computer education program. The school board member, on the other hand, requires concrete rationale for such a program and a periodic update on the status of the program.

The Troy School District, a middle to upper-middle class suburb of Detroit, recently joined the ranks of other school districts by incorporating computers into its K-12 curricula. The Troy School District includes twelve elementary schools, four middle schools, two high schools, and services 11,456 students. While Troy's K-12 computer program has existed for only about two years, it has become recognized as a model for other school districts who are just starting out, as well as for those who have had computer education programs for a period of time. The success of the Troy program can be partly attributed to the enthusiasm of the Troy staff, administration, and the community, but the single most important reason for its success is the inservice training which has accompanied each phase of development.

While undefined at that time, the training process began prior to the purchase of any computer equipment. The school district organized a small committee of teaching staff and administrators who were interested in computer education. This committee visited other school districts which had such programs and consulted the county Intermediate School District (ISD). The Interactive and Instructional Computing Department (IICD) of the ISD was able to provide much of the information the committee initially required, primarily as a result of a pilot project which had already been set up in one of the Troy elementary schools using terminals linked to their mainframe computer. After this initial investigatory work, the committee recommended to the central administration that a few microcomputers be purchased for each school. Despite hard economic times and budget cuts, the administration and school board responded favorably to the request, recognizing the need for computer literacy and skills for Troy students. The committee then spent a considerable amount of time designing a comprehensive inservice training plan that addressed the needs of administrators, teachers, and school board members.

INSERVICE TRAINING FOR ADMINISTRATORS

Inservice training is most often associated with teacher training. The Troy School District, however, recognized the key role that the administrator must play in computer education and scheduled an appropriate inservice training session for this group first. We have found that the overall support of the administration is vital to the initial success of any new school programs. Administrators need:

1. To have a basic understanding of educational applications for computers.
2. To provide assistance in the management of computer hardware and software.
3. To assist in the coordination of teacher inservice training in computer education.
4. To offer continued support to teachers so that computer education may expand and improve.
5. To communicate with the school community on how computers have been wisely used within the school.

The Troy School District began to meet these needs first by providing a half-day workshop for all administrators. Each administrator within the district was required to attend one of the three workshops presented by three Troy staff members, representing elementary and secondary levels. The presenters were all microcomputer owners and advocates of computer education within the school district. The following items were on the agenda for this workshop:

1. Overview of computer hardware
2. Rationale for computer education
3. Hardware management techniques
4. Precomputer activities
5. Software sources and concerns
6. Software examples by grade/subject area
7. Future inservice training plans

This inservice training placed great emphasis on the importance of computers within society and how these societal needs have placed a responsibility on educational systems to provide students with computer literacy and skills. The hardware management techniques focused on scheduling procedures that could be used to coordinate the sharing of a few microcomputers within a building. The most convincing phase of this workshop was the sharing of quality software examples by grade level and subject area. These examples pointed out the unique capabilities of the computer as a classroom tool. The philosophy of finding and/or creating software that directly relates to the existing curricula was stressed in preference to changing curricula in order to adapt to available software. The administrators also received information on how the computer could be used to assist them with their routine business and administrative tasks: attendance, word processing, inventory, budget, and data base management. The workshops gave administrators an excellent insight into the importance of computer education and were important in getting the program off to a good start.

STAFF INSERVICE TRAINING

Certainly one of the most practical inservice training programs is that given for the instructional staff. Teachers require detailed information on how a computer might be used in their classrooms. Teachers need:

1. To be reassured that computer education is not intended to be a replacement for teachers.
2. To know that computers are excellent tools which can embellish classroom instruction.
3. To be reminded that society utilizes computers in almost every type of business.
4. To know that research indicates that some computer skills and/or awareness will be essential for students in order for them to be employable in the near future.
5. To see actual software examples that directly relate to the current curricula.

6. To see that computers can be used as both the vehicle of instruction and the object of instruction.
7. To see how the computer can be used to create simulations within the classroom that are too costly, difficult, or dangerous to do any other way.
8. To see that the computer can be an excellent classroom management tool.
9. To have the opportunity to learn how they can modify or write their own educational software.
10. To receive additional inservice training as the school district purchases new computers and/or peripherals.
11. To see that computers can provide immediate feedback to students. This is often difficult to do in the regular classroom setting.

It should be noted that the introduction of computers into the curricula will not be welcomed or accepted by all staff members. Some teachers will view computers as a threat to their jobs and others will accurately indicate that time does not exist in lesson plans to accommodate such "add-ons." Our experience has shown that the best way of dealing with these problems is to first offer staff inservice training to only those who are interested. The majority of the remaining teachers quickly realize the benefits and rewards that computers offer as they observe the teachers and students who do use them.

The first inservice training offered to the Troy staff was a one-session overview of computers and how they could be used in the classroom. The fact that these sessions were voluntary is a primary factor in their success as described earlier. The workshop was presented three times--once for elementary teachers, once for middle school teachers, and once for high school teachers. The agenda for these sessions was identical to the one used for administrators. Emphasis, however, was placed on different topics. The teachers received an expanded section on software examples by subject and grade level. This was not difficult to accomplish since the audience for each session was grouped by one of the three levels mentioned earlier. The teachers also received more examples of precomputer and literacy activities. Attendance was moderate: elementary 18.3%, middle school 25.0%, and high school 12.0%. These teachers provided the catalyst of enthusiasm which helped the program to get off to a good start.

Shortly after the first inservice training session, each elementary school and middle school received three microcomputers. The high school Math Department received five microcomputers. The elementary schools purchased PETS and the secondary purchased APPLES. A small assortment of software was also purchased for each building. A second inservice training session was organized to provide staff members with hands-on instruction on how to operate the equipment. This training session was offered several times in the district and each session was quickly filled. To accommodate other interested staff members, the media specialist from each building attended the inservice training and now serves as a resource person within each building. The agenda for these workshops was as follows:

1. Care and maintenance of the microcomputer
2. How to set up the microcomputer
3. Software handling techniques
4. How to load and run software
5. Simple microcomputer commands

The third inservice training offered to teachers covered how to write educational software. This has been one of the most popular inservice sessions ever offered to Troy teachers. Those who sign up for this workshop attend six two-hour sessions. Teachers are not charged a fee to take this class, nor do they receive any compensation or graduate credit for this, although some secondary teachers have expressed an interest in such rewards. Classes are offered after school hours and on Saturdays at a convenient location in the school district. Each session that has been offered has quickly filled to capacity, with a waiting list for future sessions. Elementary teachers have been particularly interested in these classes. To date, 62.4% of the elementary and 11.8% of the secondary staff have taken this training. One might stereotype that it is the younger teacher who might take such a class, but this is far from true. Teachers with thirty years experience as well as the younger teachers attend. Administrators are frequently enrolled also. Interest has been sufficient that an advanced programming class has been added for interested elementary teachers.

The school district has been able to purchase a few peripherals for the microcomputers. Disk drives and printers have been purchased for each elementary school. As these items were purchased, inservice training was provided to ensure the equipment was properly connected and operated.

In order to provide frequent communications to staff on computer education and upcoming inservice sessions, the school district now publishes The Troy Computer News. This newsletter is written by a group of teachers and administrators representing each discipline that utilizes computers. Articles include information on inservice training that is offered within the district, conferences held outside the district, hardware and software concerns, recent computer purchases, and the status of computer education within the district. Feedback from staff indicates that they would like additional newsletters in the future. Other school districts have requested to be on the mailing list in order to stay abreast of what is happening in Troy.

SCHOOL BOARD INSERVICE TRAINING

School board members are naturally interested in and responsible for providing direction and leadership to be overall instructional program. Today's economy dictates that each school board member must carefully evaluate the need for and productivity of any program. The fact that computer education is emerging during times when program cutbacks are occurring places an added need for regular review and evaluation of the computer program on board members. Computer education requires a relatively high expenditure from the district budget. School board members need:

1. To be assured that computer hardware will be well used within each school.
2. To know that computers are not another fad expenditure that will soon collect dust among the archives of educational mistakes. (Remember language labs?)
3. To see that learning with the aid of a computer is not a sterile experience. Teacher interaction and supervision are key elements to the student's experience.
4. To become computer education literates in order to provide community members with rationale and support of the program.
5. To know that a computer education program is designed to prepare students to take an active and productive role in society.
6. To see that computer literacy is a vital part of education as demanded by society.

The Troy School District meets these needs by providing regular updates to the board of education during board study sessions. The board has made specific requests, as a result of these inservice sessions, to receive recommendations for the purchase of additional equipment that would satisfy a total computer-assisted curriculum. This continued support has made it possible for the Troy computer program to continue to grow. Two computer education updates have become a regular part of the annual school board agenda. One session focuses on the status of the elementary computer education program while the other is secondary oriented. The Troy administration views these sessions as imperative in order to maintain the high level of support the board has given to computer education.

An off-shoot of this phase of inservice training is the periodic offering of community demonstrations on how computers are being used in the Troy Schools. These demonstrations are frequently presented at parent-teacher meetings, science fairs, and during school open houses. One school even uses the computer to display students' stories during a Young Authors Conference. The Troy community, like all communities, is seriously interested in seeing how its tax dollars are spent. These demonstrations do an excellent job of showing the broad applications and use of our equipment. One remarkable result of these demonstrations is the fact that many parent organizations have shown total endorsement of the program by purchasing additional hardware and software for the schools. Further, many parents have volunteered to work in the schools to provide assistance and supervision to students as they use the computers.

SUMMARY

The importance and usefulness of computer education has been well documented. Society has already found a multitude of applications for computers, and these applications grow at an ever-increasing pace. Now that society has placed such an importance on computers, education has the responsibility to prepare students for such a society. Microcomputers have made it economically feasible for educational institutions to provide computer literacy and computer skills to their students. A computer education program, however, does not exist by simply purchasing an array of hardware and software. Administrators, teachers, and school board members all need to receive quality inservice training in order to properly utilize such equipment as well as to appreciate and recognize the applicability of computer-assisted instruction in the educational curricula. Inservice training should begin before the first switch is turned on, complement each phase of development, and be a continuing responsibility of educators. The Troy School District has recognized these needs and has been successful in implementing an effective on-going inservice training program. This training is respected as the primary reason the computer education program in Troy started efficiently, has gained a wide base of acceptance, and will continue to grow in the future.

A TEACHER'S INTRODUCTION TO COMPUTERS

William J. Taffe, Professor of Physics
and
Stephen M. Weissmann, Associate Professor of Psychology
Plymouth State College
of the University System of New Hampshire
Plymouth, New Hampshire 03264
(603) 536-1550 Ext. 321

INTRODUCTION

This paper describes the first workshop, in a series of workshops, for elementary and secondary school teachers who are interested in introducing computers into their classrooms. We present the goals of our program, its methodology, and the evaluation by the participants.

The problem we approached is undoubtedly common throughout the U.S. Teachers at the elementary and secondary levels are developing an awareness that they can improve their students' education by including computers in their teaching and they are seeking help in getting started. The precise nature of what they hope to accomplish, however, is not clearly formulated. Questions they frequently asked us were of the nature: "Which micro would you suggest we buy?" or "What do you think we should be teaching?". Their uncertainty was more often "felt" than precisely expressed but, clearly, teachers were communicating a hope that we might somehow "help".

Our geography, a rural setting in the northeast, made the teachers' problems a bit more severe. In more populated regions, computer clubs, computer stores, and nearby teachers using computers would provide teachers with more opportunity to assess goals, methodologies, and equipment, and to share and swap ideas with others in similar situations. Plymouth State College is the only institution in the region which might provide this assistance.

Our Computer Science program is in an evolutionary state and not in a position to offer formal assistance. The New Hampshire Association for Computer Education Statewide (NHACES) is still in its formative stages and is predominantly anchored in the southern, more populated, region of New Hampshire. So, the authors, in co-operation with our college's Division of Continuing Education, voluntarily developed a series of workshops for teachers designed to share the ideas and knowledge about teaching and computers which we have gained through the use of computers in our own teaching, reading, and from our wives' knowledge and current experience in pre-college education.

Our goals were to expose teachers to basic information about computers, to stimulate their interest in teaching with computers, and to reduce their anxiety about the process. We began by developing a detailed plan for an initial workshop and potential topics for subsequent workshops, and then solicited reactions from several school superintendents. Based on these, we set a date for the first workshop and advertised it through the continuing education program. The response was overwhelming. The first workshop was filled within a matter of days and several more dates were chosen to meet the requests which poured in. We offered the first workshop in October 1981, and are still offering sessions of it. The more advanced workshops began in late winter.

The logistics of the workshops are not unusual. They are held on Saturdays for three hours in the morning and continue as long as teachers wish to stay in the afternoon. Each is limited to 15 participants and the fee is $40. The first workshop provides a foundation "for the teacher who needs to start from scratch", on which we can later base more advanced workshops on such topics as "BASIC programming", "CAI" and "Microcomputer evaluation and selection".

THE INTRODUCTORY WORKSHOP

The first three hours of the workshop consist of lecture, discussion, and demonstrations by the two instructors.

Hardware

We begin with a very brief introduction to the concepts and terms of computer hardware: the CPU, RAM and ROM, secondary storage and I/O devices, illustrating these concepts with actual hardware. The cover is removed from an Apple and various components are displayed. A cassette player is connected and a tape is read in. While the tape is being read, we demonstrate a disk drive and diskettes, and compare the two formats with respect to factors such as speed and cost.

Originally, we had intended to discuss this technical information later in the presentation. However, the experience of the first workshop revealed that this is basic information that the teachers continually asked about while we were attempting to develop concepts of computer use in the classroom. Leading off with this information seems to clarify the teachers' understanding by giving the group a common set of terms and concepts, and it actually saves time by forestalling many questions which would interrupt the flow at other points in the workshop.

Computers in the Classroom

We then move to the topic "Computers in the Classroom". This topic was placed early in the workshop for two reasons. First: teachers tend to be rather pragmatic and frequently seek applications-oriented knowledge. In such situations, application followed by theory is pedagogically more effective than the reverse approach. Secondly, the examples available in this section generate a sense of excitement which sets a tone for the entire workshop.

Computers in the Classroom are examined in three categories: teaching with computers, teaching about computers, and computers in classroom management. Teaching with computers is begun by asking who might be taught by computer. Everybody? Special needs students (gifted and/or handicapped)? This leads to a discussion of how the computer will be used. As a primary instructional vehicle? As a supplement to other instruction? We then demonstrate software which can assist teaching through various techniques. As we demonstrate each program, the leaders and participants discuss the pros and cons of the technique and how it might be implemented.

The techniques discussed are: drill and test (demonstrated by "Darts" and "Don't Fall" found on Elementary, My Dear Apple), tutorial ("Legends" from the Apple Pilot Diskette), simulations ("Lemonade: from Elementary, My Dear Apple), an expert for the classroom (using "Tellstar" which displays, identifies, locates, and draws constellations for any date, place, and time in the Northern Hemisphere), and learning by teaching or 'the computer as tutee' (using the "Animals" game from the Apple System Master). We emphasize that participants should choose techniques and approaches which fit their situation and circumstances. We stress that there is no 'right' technique. Instead, the teacher is being given some new teaching tools and has the freedom and responsibility to make appropriate choices. Likewise, we stress that computer-aided learning is not a panacea for all of education's ills but a potential technique for improving some aspects of instruction.

Teaching about computers is postponed until after the teachers have been exposed to a bit more explanation about the way in which computers work and how they process information.

We then take a brief look at computer-assisted classroom management: the computer as a bookkeeper (using "Apple Grade Book" as an example), a calculator for complex mathematical problems, and a text editor. Although we mention the use of computers for school administrative functions, we place this outside the scope of "a teacher's workshop".

Development of Computer Concepts and Software

Following a short break we briefly trace the history of computer concepts, emphasizing that electronic computing machines are new but the problems they were designed to solve are ancient.

The nature of how information is represented and stored in the memory of the computer is developed in an innovative and apparently effective manner. Using the SRA Discovery program, the class learns to code Hollerith cards. Each teacher receives a set of blank cards and a copy of the Hollerith code. While the codes for various letters and numbers are displayed on the screen, typically, each teacher codes his or her name on the blank cards. The spaces are filled in with pencil to correspond to the holes of the punched card.

From this exercise it is relatively easy to introduce the concepts of the binary digit (bit) and sequences of: switches off or on, voltages of 0 or +5 and 0's and 1's. Byte and word organizational structures are then introduced. Having just represented their own names as patterns of filled and unfilled spaces on Hollerith cards, these new concepts are familiar rather than threatening.

This leads directly to the need for higher-level computer languages and the major ones are identified. The problems of inter-machine compatability and dialectical differences are discussed.

We conclude this section with a discussion of how and where to get more information. Upon arrival each teacher had received an extensive packet of information, including a detailed glossary of computer terms, several examples of reviews of computer books and software packages, an annotated bibliography of resources, and copies of advertisements from current computer magazines. We review this material with the teachers and have a general discussion of where to go next. Frequently this evokes a sharing of personal experiences with specific sources (stores, brands, etc.) by members of the group.

Finally, a handout with a comparison of four different computer systems is distributed. The various features of the AppleII, Atari 8000, CBM PET and the TRS-80 III are compared. We are careful to emphasize at this point that the determination of the school's needs and plans should come before the decision to purchase specific equipment.

How Computers Solve Problems

The goal of this portion of the workshop is to show how computers process information (solve problems). We introduce the conceptual bases of programming: notions such as sequencing, assignment, loops, test and control, I/O, etc. We emphasize that the task and logical sequencing of tasks, not the syntax of a language, are the essence of programming. A simple CAI program is then flowcharted to show programming concepts in action and this is followed by a viewgraph of that flowchart coded in the PILOT language.

Teaching About Computers

The participants now have a good (although brief) overview of how computers 'do their thing' and how a computer might assist them in teaching. We then address the topic teaching about computers. This is subdivided into two categories: computer awareness and data processing. The latter is presented as a program most frequently encountered in business or vocational education. We introduce computer awareness by discussing what it means (how computers work, how they are controlled, i.e. programming, and their use and impact on society) and proceed to give examples of how this might be accomplished either in one specific course or little-by-little throughout the curriculum. Emphasizing that it requires very careful planning and coordination, we describe a scenario wherein students could be introduced to the various aspects of computers via their regular academic subjects from primary school through senior high. Strategies are briefly examined and we suggest that while we can learn from the experiences of others, at present, there is no 'right way' to approach such a developing and expanding phenomenon. We end this portion by handing out a set of suggested competencies, simplified from those given by the ACM, which might be appropriate goals for a teacher who wishes to become computer literate.

Evaluation and Hands-on

Throughout the workshop both instructors are actively involved. We complement each other's presentation, not only by anticipating needs (handouts, setting up demonstration programs, etc.), but also by interjecting comments, spotting questions or difficulties, and in general participating in the delivery of the other's information. This collegial interaction has been particularly cited by the participants as a significant feature of the workshop.

After a short break to complete the evaluations, we turn to the hands-on component. Individually and in small groups the teachers are encouraged to interact with and explore the programs which they have seen demonstrated and others we have prepared. These are available on three Apples in the classroom and on the college's PDP 11/70 accessible from a nearby terminal cluster. Rather than just demonstrate the software, the instructors encourage the teachers to actively experiment with the machines and the programs.

Although varying with the individual teacher's interest, the most popular programs come from the Elementary My Dear Apple disk: "Darts" (a fraction exercise in the form of a game) and "Don't Fall", a variation on the hangman spelling game. Also popular is "Animals", a program from Apple's System Master, which allows the user to teach the computer to distinguish animals.

The degree of involvement varies with the individual teacher. Some devote most of their time to playing with the programs as games, others to assessing pedagogical value, while others are more interested in working with the hardware - handling the disks, exploring control features, etc. These teachers are likely to catalog the disks and to examine listings of the programs. Another group has seemed more interested in programming and we often develop the rudiments of Basic in a few minutes with these teachers.

The instructors are available for several hours after the formal part of the workshop. The class is encouraged to break for lunch and return to interact with the software. The teachers are also encouraged to return to the campus at a later date. They each receive an account number on the PDP 11/70 and instructions on how to log on. Other teachers have returned to explore the programs available on the Apple. This is by individual arrangement, since the microcomputers are ordinarily housed in their academic departments.

EVALUATION OF THE WORKSHOP

Before we turn to the hands-on component , we ask the teachers to fill out an evaluation of the workshop. At the first workshop we sent the form home with the teachers with a stamped, self-addressed envelope, hoping to elicit more considered responses. However, the return rate was less than 20%. Administering the form at this point yields a 100% response.

The evaluation data derive from three distinct sources: answers to the objective questions on the evaluation form, narrative responses to questions on the form, and anecdotal comments made during the hands-on period and after the workshop. Each of these will be discussed.

Objective Responses

The objective data reveal an extremely positive reaction to the workshop. When asked to rate overall satisfaction on a 5-point scale (1 = not satisfied to 5 = very satisfied), 8 teachers chose 4, and 24 of them chose 5. Perhaps more importantly, when asked to indicate the likelihood of seeking more information, 30 of the 32 respondents chose 5 (very likely). From the standpoint of stimulating interest in computer uses, the workshop appears to have been very successful.

Narrative Responses

We also asked the teachers to indicate specific aspects of the workshop which they did and did not like. Responses ranged from the global ("Everything was worderful!") to the specific ("I don't like stale donuts"). The positive aspects included: organization of information, avoidance of jargon without oversimplification, stimulation of interest and, perhaps most often, the success of the team approach. Negative comments were almost non-existent.

Two comments may best reveal the success of the workshop. "Excellent -- I hope there will be more. P.S. The $40 came out of my pocket." Apparently we are providing more than just another routine exercise to satisfy continuing certification requirements if teachers will assume the expense themselves. And, "When I got home there was a newly arrived Radio Shack flyer and it felt good to look at the computer ads and have some idea of what they were talking about". It would seem that we achieved our objective in this introductory workshop.

Anecdotal Response

Throughout the afternoon we engage the teachers in discussions of their particular interests and needs. Their comments range from specific questions about software for their classes to questions of, "How can I learn Basic?".

One of the most common questions is, "Will you come to my school and talk to more of us?". Overwhelmingly we sense that these teachers are sincerely motivated to learn more about computers and to apply this technology in their classrooms. While many of their comments are enormously satisfying to us as instructors they also reveal an enormous demand for more information on the use of computers in education.

THE FUTURE

The evaluation form also provides the opportunity for the teachers to indicate how they would most like to pursue more advanced information about computers. They are first asked to choose up to 3 different formats from a list of 6. The results are presented in Table 1. Almost every respondent expressed the intention of taking more workshops and even a series of workshops leading to a certificate of literacy. About half of the teachers indicated a preference for a formal college credit course. The interest certainly seems to exist.

Secondly, they were asked to select up to 3 topics for more advanced workshops from a list of 7. The results are presented in Table 2. The majority of teachers indicated a preference for workshops on Basic, CAI, and software selection, although every topic was selected by at least 25% of the participants.

TABLE 1
PREFERENCES FOR SEEKING
ADDITIONAL INFORMATION
32 Teachers; 94 Responses

NUMBER OF TEACHERS	FORMAT
29	Attend a series of workshops for a certificate of computer literacy
27	Attend a more advanced workshop
15	Take a college course for credit
10	Read books or magazines
8	Talk to computer salespeople or professionals
5	Take a series of college courses for a certificate or degree

TABLE 2
PREFERENCES FOR ADVANCED WORKSHOPS
32 Teachers; 102 Responses

NUMBER OF TEACHERS	TOPIC
22	'Basic' programming
20	Computer-assisted instruction
18	Commercial software selection and evaluation
12	Microcomputer selection and evaluation
10	Classroom management
9	Wordprocessing
8	Teaching about computers

Based on the reactions to the first workshops and the perceived demand for more information, we are responding in

2 ways. First, additional introductory workshops will be announced and we will explore delivering workshops in the school districts themselves, rather than at the college. Second, we are proposing a Certificate of Computer Literacy, to be awarded for successful completion of 5 workshops.

To further this end we are developing 3 advanced workshops to be delivered in late winter of 1982: Basic Programming, CAI, and Microcomputer Evaluation and Selection. We plan to retain the Saturday morning format, most likely offering the introductory workshop followed by a sequence of the advanced workshops. The cycles will be repeated for as long as an audience exists. Two more topics will be developed for the summer, and by the fall of 1982 we plan to run complete cycles on a continuing basis.

We are also encouraging the teachers to consider the courses offered by our Computer Science Department in the summer. Since almost one-half of the particpants expressed an interest in taking college courses, we consider them likely candidates for courses in CAI, Basic programming and others.

What have we learned from this experience? First, we have confirmed that public school teachers have a great thirst for information about the computer and its applications in their classrooms. Second, we have demonstrated that two instructors without computer science degrees, but with a lot of experience using the computer, can deliver an introduction to computers in education in an informative, friendly fashion that has been extremely well received. Third, we have developed a model for delivering workshops and perhaps even college courses to teachers that appears to be very successful. We have found this teaching experience to be extremely rewarding and personally satisfying.

Pre-College Computer Science

The Duke University Computer Kamp

Kevin Bowyer
Computer Science Department

Mel Ray
Vice Chancellor for Data Processing

Gary Laxer*
Computer Science Department
Duke University, Durham, North Carolina 27706

INTRODUCTION

During the summer of 1981 Duke University made the initial offering of the Duke University Computer Kamps (DUCK). The educational objective of DUCK is to provide young people a fundamental "computing literacy." "Computing Literacy" as defined by DUCK is an appreciation of the general principles which underlie computer hardware, software, and the application of computing technology to various science, business, education, government, and entertainment objectives. This is distinct from the goal of simply creating programming competence in a certain language or the goal of creating familiarity with a particular type of computer.

The 1981 DUCK included two-week residential sessions for each of the age groups 16-17, 14-15, and 12-13 years old, and a one-week nonresidential session for 10-11 year olds. Each session accommodated sixty kampers, divided into two groups of thirty kampers each. Two hundred thirty-six (236) kampers received certificates of completion for participating in the 1981 DUCK experience. The purpose of this report is to describe the structure, content, and success of DUCK.

FACILITIES

The residential kampers were housed in Duke undergraduate dormitory rooms and ate their meals at the same cafeteria that serves regular Duke students and faculty during the summer.

A microcomputer lab facility was provided to DUCK through the courtesy of Radio Shack, a division of the Tandy Corporation. Forty (40) TRS-80 Model III personal computers and a vast library of software, disks, tape cassettes, and manuals made up the computer lab. Twenty (20) of the forty TRS-80 computers were 48K double disk systems, ten (10) were 16K cassette-based systems, and another ten were color computers. A large lecture hall was renovated for the summer to accommodate the computer lab (figure 1). In addition to the special computer lab, some of the advanced kampers made use of the computing facilities regularly available to Duke students (IBM 5100 personal computers and the IBM 370 systems available through Triangle Universities Computation Center). Other DUCK facilities included regular use of a classroom (figure 2) and occasional use of a videotape viewing room available in Perkins Library on the Duke campus.

ACADEMIC PLAN

The sixty kampers in each session were divided into two teams of thirty each for instructional activities. Each team was assigned a lead instructor who coordinated their educational experience for the entire session. The educational experience consisted of classroom lectures with the instructor, laboratory periods with the personal computers, classroom discussions with a variety of guest speakers, and field trips to facilities beyond the Duke campus.

The DUCK classroom lectures and laboratory periods were designed to provide kampers a basic technical and theoretical introduction to computing literacy. Early in the first week the emphasis was on understanding the "Von Neumann machine" model of computer operation. These concepts involve a general understanding of computer hardware organization and "how computers work." This information became the framework for understanding modern computers. During this time the kampers wrote their first computer programs. The programming was done in the microcomputer lab where each kamper had access to a personal computer. The instructor outlined and explained the program at the front of the class, while lab assistants circulated among the kampers to answer specific questions and explain error messages. The "one student-one computer" setting was very successful. With an instructor and adequate lab assistants, this format provided a very tight feedback loop and a near optimal learning environment. The frustration level of the students was greatly reduced over more traditional methods of teaching programming concepts. The "lecture about programming concepts - send students off to program" method of doing things is hopelessly inferior to this method. The only drawback of the lab oriented approach to computer instruction is that it is both labor and hardware intensive.

---

* Now with the Rose-Hulman Institute of Technology in Terre-Haute, Indiana.

Near the middle of the first week, all kampers were given an individual project assignment which they completed during the remainder of the week. Examples of individual project assignments include 1) a "text processor" program that counts the number of characters, words, lines, paragraphs, and pages typed, and 2) a statistical summary program that sorts a list of numbers and computes the mean, median, and mode. These program assignments were quite ambitious for kampers who have been programming for just a few days, and were embellished to the extent that they challenged most of the more experienced kampers. In addition, the project assignments required the working programs be accompanied by a page or two of documentation which described the purpose of the program, the variables and data structures used, the way in which the problem was solved, and the test cases which were run to check the program.

At the beginning of the second week, the kamper was introduced to software engineering and more detailed problem solving. As the individual projects were completed, each group was organized into teams of three or four kampers for a team programming assignment. The team program projects were generally not much more ambitious than the individual projects. However, the additional complexity of organizing a team and co-operating in the solution of a problem made this a very ambitious learning experience. The idea of working together in teams and co-operating to develop an integrated computer program was something very new and challenging to all age groups in the kamps. This technique was an excellent learning experience beyond just the computer work involved. The team and individual projects were coordinated so that some projects gave emphasis to numerical processing and others emphasized character and logical processing.

Virtually all of the programming efforts of the kampers were completed on personal (desk top) computer systems. The language used for nearly all the programming was Basic. No assembly language programming was attempted in the 1981 DUCK. The Kamp faculty regard the choice of a programming language as a secondary choice, and mainly one of convenience. The objective was to use a language that can be learned very quickly, so the kampers could be programming as soon as possible. In addition, Basic is often the best supported language for microcomputers. While more academically oriented languages were available, Basic allowed the kampers to begin programming on the first day with very little learning overhead.

A typical daily schedule during the 1981 DUCK might look as follows:

8:00-9:00	Breakfast
9:00-10:15	Classroom session
10:15-10:30	Break
10:30-11:45	Programming lab
11:45-1:45	Lunch and Recreation
1:45-3:00	Guest speaker
3:00-3:15	Break
3:15-4:30	Programming lab
4:30-6:00	Dinner
6:00-7:00	"Easy living"
7:00-9:00	Social activity or time in computer lab

The objective of the guest speakers was to provide a broad spectrum survey of computer systems applications. These discussions allowed the kampers to gain insights into how computer systems are used in our society. Many of the guest speakers were not primarily technical people, but rather people in charge of the use of computers in different organizations. Over the four sessions of the 1981 DUCK, guest speakers discussed the following topics:

- Computers and Government
- Computers and Space Exploration
- Microelectronics and VLSI
- Computers and Marketing
- Computers and the Fine Arts
- Computers and Law Enforcement
- Regional Computing
- Computers and Health Care
- Computers and Sports

At least one afternoon a week the regular activities would be pre-empted for a field trip. At various times during the 1981 DUCK, field trips were taken to

- IBM at Research Triangle Park,
- Burroughs Wellcome (a pharmaceutical research lab) at Research Triangle Park,
- North Carolina Museum of Life and Science,
- Triangle Universities Computation Center at Research Triangle Park,
- Ultrasound Laboratory in Duke Hospital,
- Morehead Planetarium at University of North Carolina at Chapel Hill, and
- Duke University Computation Center.

The purpose of the field trips was to allow the kampers to observe important applications of computers "in situ". The field trips were consistently one of the more popular activities with the kampers. This activity augmented the guest speakers in terms of providing the kampers a perspective on how computers affect our society.

RECREATIONAL ACTIVITIES

On evenings and weekends, the kampers had a full schedule of planned recreational activities. They made regular use of the aquatic center, tennis courts, and gym on Duke campus. Other activities included movies both on Duke campus and at a local theatre in Durham; roller skating at a roller rink in Durham; and an opportunity to see a Durham Bulls semi-pro baseball game. In addition, the Kampers enjoyed pizza parties, ice cream parties, cookouts, and picnics.

DISCUSSION

The first DUCK experience was successful beyond the expectations of anyone involved. This is true both in terms of public interest (media coverage) in the program and in terms of what could be taught in a short time to young people 10-17 years of age.

More than 300 applications for admission to DUCK were declined after the 240 openings in the 1981 DUCK were filled. Interest ran strongest in the 12-13 year old age bracket, followed closely by the 14-15 year old age bracket, then the 10-11 and 16-17 year old age brackets. The percentage of female participants in DUCK tended to increase with the age bracket, to the point where the 16-17 year old kampers were almost evenly distributed by sex. The 1981 DUCK program drew participants from throughout the eastern seaboard and the southeast. Participants from as far away as New York, New Jersey, Florida, and Texas were in each session. International participants included a resident of Mexico and another from the Netherlands, both of whose families were visiting North Carolina at the time their kampers attended DUCK.

Program assignments of roughly the same difficulty, and sometimes exactly the same assignments, were given to beginning students in each of the 12-13, 14-15, and 16-17 year old age groups. It is the faculty's general impression that the 12-13 year olds could cope as well intellectually with the program assignments and classroom concepts as could the 14-15 and 16-17 year olds. There was no planned prospective study, and DUCK certainly makes no claims to a "representative" sample of any well-defined population. However, the DUCK experience does indicate that computer science can be successfully taught to age groups much younger than is normally done.

It was necessary that expectations be scaled down for the 10-11 year old age group, if only because their session was just one week in length. They were successful at scaled down versions of the individual project used for older age groups, but required noticeably more coaching through the solution.

The academic staff for DUCK began as two instructors and three lab assistants. This allowed for an average of three instructional staff in the computer lab with one group of thirty kampers. It was found that this staff/kamper ratio was too small and eventually three more lab assistants were added for a ratio of one instructional staff to every five or six kampers in the lab. Most of the lab assistants were Duke undergraduate majors in computer science who were already known to one of the instructors.

The DUCK administrative staff included four residential advisors (counselors), a recreational activities coordinator, and a kamp administrator to oversee daily kamp operations. The counselors lived in the dorms with the kampers and provided after-hours supervision. The kamps operated under the Duke Summer Programs Office, Dr. Robert Sawyer, Director.

The subject of "computer games" must be addressed by an activity such as DUCK. Most personal computers have available a wide range of computer games. The most popular games fall into two categories - fantasy games such as the "Adventure" series and science fiction games which deal with spaceships, targets, and weapons of some type. These games can be viewed as psychologically addictive and many kampers of the age groups we observed would often miss meals to play them. This phenomenon was managed by asking that the kampers play games only in the after-dinner (optional) lab sessions, or only after completing certain milestones in their assigned projects. One period was reserved for a computer games tournament to identify a "computer games champion". In general these measures were sufficient to keep the level of game-playing under control, though there were always some individuals who required more guidance than others.

FUTURE PLANS

Only a relative handful of computer kamps were offered during the summer of 1981. To our knowledge the earliest program that might be called a "computer camp" occurred in the summer of 1979 [1]. However, our informal discussions with other universities around the country indicate the number of computer camps may be following an exponential growth path.

Several changes are planned for the 1982 version of DUCK. The capacity will be doubled to handle approximately 500 kampers. The kamps will be organized by age group, however each Kamp should have both novice and advanced tracks for some age groups.

The 1982 DUCK will operate using the new IBM personal computer systems to equip the lab facilities. The curriculum will have only relatively minor changes. Less time will be allocated to programming per se, and more to the use of package programs such as text processors, VISICALC, and other "canned" programs. The daily schedule is modified so that classroom sessions are shorter in length. Another change being considered is an alteration in the ratio of guest speakers to field trips.

The Duke University Computer Kamp was created and organized by Mr. Mel Ray, Vice Chancellor for Data Processing, and Dr. Kevin Bowyer, Research Assistant Professor in the Department of Computer Science.

REFERENCES

(1) Wall Street Journal, July 30, 1980. "Dear Mom: At Camp Today I Swam, Rode, and Ran a Computer".

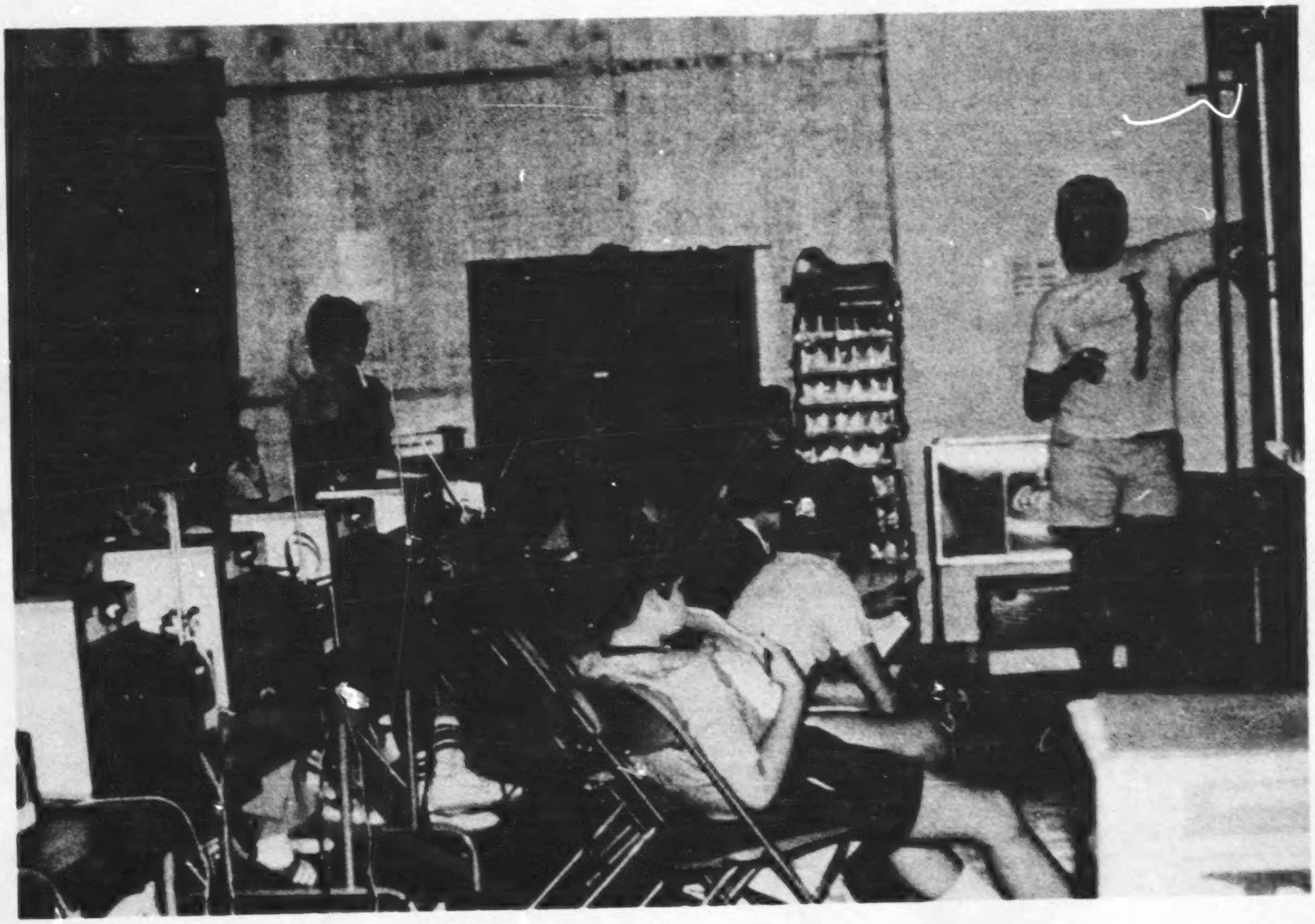


Figure 1 - The DUCK Laboratory

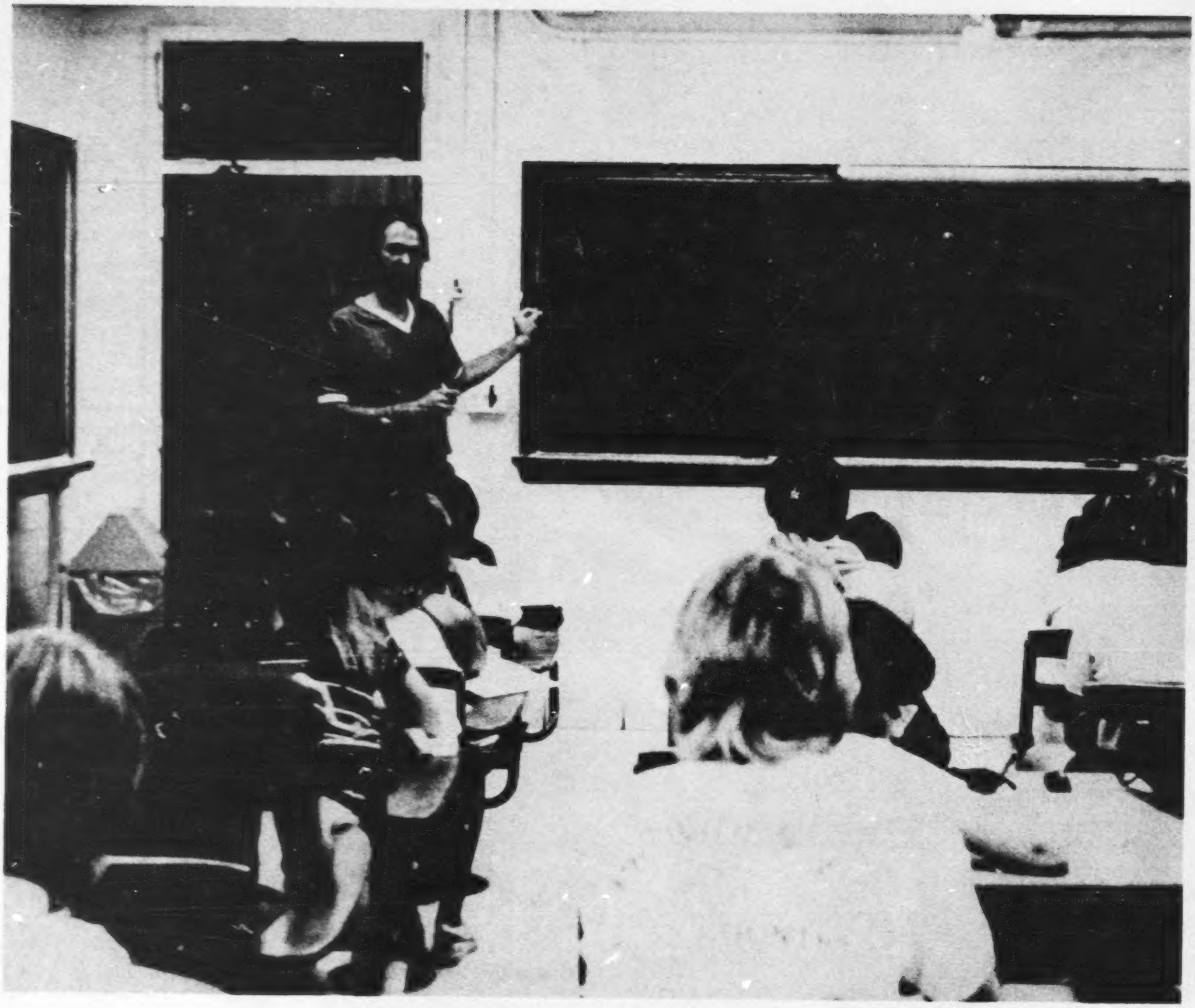


Figure 2 - The DUCK Classroom

COMPUTER EDUCATION AT MORELAND ELEMENTARY SCHOOL--THE FIRST YEAR

Dr. Beth Lazerick
Shaker Heights City School District

Computers and computer education are making an impact at Moreland Elementary School in Shaker Heights, Ohio. While in most areas computers are relegated to only a few students at the high school level and to some in college, Moreland students are participating in a comprehensive program of computer education.

Moreland is a K-6 elementary school which was selected to host a magnet program in computer education beginning in September 1981. A coordinator of elementary computer education was hired to lead this program in addition to working throughout the system in the area of computer literacy. The school has a computer education lab equipped with 17 TRS-80 Model III's and one Atari 800, two printers and three disk drives. The TRS-80s are connected through a Model 2 network system that enables the teacher to upload from a master computer. Every regular classroom in the school houses a TRS-80 or an Atari 400 or 800 for day-to-day use. The library media center has three TRS-80 computers and the Discovery Room, another special feature of the school, has two Apple computers with disks and a printer.

The Moreland program is unique in several ways: First, all children in the school have the opportunity to use the computers for instructional purposes. This occurs in their own classrooms and in the computer lab. All children also have the opportunity to learn the basics of computer programming beginning in the first grade. Granted, during the early years children are not really learning to program computers. However, they are becoming familiar with the technology, terminology, and the keyboard. Secondly, Moreland offers typing courses to children in grades 3-6. Younger children are given introductory work to familiarize them with the skills involved in typing. Thirdly, all Moreland teachers have had some inservice training and currently have some skills in computer programming.

Children in grades 1-3 receive a minimum of one-half hour of instruction from the coordinator of elementary computer education. Children in grades 4-6 receive two half-hours of instruction. Some teachers do more work with additional topics within the normal school day and after school. Children are also invited to join a games club that meets during lunch hour and, in grades 3-6, an afterschool computer club which hosts speakers and provides additional activities is also in operation. The computer lab is open at least twice a week for additional work after school.

The thrust of the program is computer literacy through the acquisition of programming skills. Computer assisted instruction is used within the classrooms and, to a limited extent, in the computer lab. The vast majority of time, however, is spent learning ways in which to make the computer behave as the programmer wants it to behave! The preliminary goals and objectives of the program are as follows:

GOALS:Computer literacy is the acquisition of knowledge, skills, and attitudes that enable a student to
A. Become familiar with the uses and functions of computers in our changing world.
B. Interact with and use computers to enhance and enrich his or her learning as indicated by needs within the school environment.
C. Program and manipulate the computer while creating simple programs to pro-

duce outputs in accordance with aptitude and interest.

D. Learn increasingly more complex programming skills and begin to create unique work that reflect individual interests and abilities.

E. Develop problem solving and logical thinking skills through the challenges of programming and of computer-assisted instruction.

F. Gain confidence and develop attitudes, values, and motivation related to the use and function of computers.

OBJECTIVES:(Cognitive)

1. Computer literacy and general understanding: The student will be able to

A. State some of the uses of computers in our lives.

B. Describe a computer and its component parts.

C. Trace the history and development of computers.

D. Acquire knowledge about how computers will interact with various life styles and proposed career choices.

E. Recognize and describe some of the advantages, disadvantages, and possible abuses of computers in society.

II. Programming: The student will be able to

A. Operate a computer, turn it on, locate specific keys, access and use peripheral devices

B. Create simple flowcharts to illustrate correct logical thinking.

C. Follow simple programs in the basic language to their logical outcomes.

D. Detect errors in simple programs and modify them so that they will run.

E. Copy, create, and adapt simple programs that illustrate the proper use of the following commands and instructions:

1. Return or Enter
2. Run
3. List
4. New
5. Break
6. Con-t
7. End
8. Rem
9. Csave
10. Cload
11. Print
12. Goto
13. For-next
14. "" "" (quotation marks)
15. "," and ";" for spacing
16. If-then
17. If - then else
18. Let
19. Input A
20. Input A$
21. Mathematical functions (+,-,*,/)
22. Exponentiation
23. For X= ___to ___ Step A
24. Rnd
25. Dim (a, 20)
26. Subscripted variables
27. Subscripted string variables
28. Gosub, return
29. On X goto
30. Plot, drawto or set, reset
31. Read, Data
32. Int
33. Abs

F. Utilize advanced techniques including the following:

1. Arrays
2. Double subscripts
3. Files
4. Editing

(Affective)

III. Throughout the study of computers the student will

A. Become aware of the uses of computers in society.

B. Show an interest in computers and their uses.

C. Feel confident with his current skills in programming.

D. Value the advantages of computerization.

E. Value the uses of computers in making life more creative and interesting

F. Choose to use discretionary time for work with computers.

G. Choose to use computers to help solve problems or to enhance learning.

H. Identify one's career or personal interest as being influenced by computerization.

These objectives will not be reached during the first or even the second year of the program. As children are part of this comprehensive study of computers, more and more children should reach many of these objectives by the end of the sixth grade. Realistically other objectives will be added as children become more computer literate and more and more homes obtain computers for personal use.

Since the beginning of the 1981-82 school year the students have been actively involved in acquiring the goals and objectives stated. Following are summaries of what most of the children at each grade level have accomplished. Each higher grade level has also been exposed to the skills mentioned for the younger grades. In these cases only additional learnings are listed.

Grade 1

The youngest children can locate most letters and symbols on the keyboard. They can produce quotation marks, question marks, as well as the plus sign which requires the use of the shift key. The concept of simultaneously striking two keys was a difficult one for these children at first. First graders are familiar with the following instructions and commands: LIST, NEW, RUN, CLOAD, PRINT, GOTO, and INPUT A. They can type one to three lines of code to produce a short program. Many children understand the loop concept and are able to ask the computer to count or repeatedly print some statement. We have talked about various means of input (disk, cassette, punchcard, and keyboard) and they are familiar with art, music, and artificial intelligence as demonstrated on the computer. These children have done CAI involving keyboard recognition, alphabetizing, math facts, and interactive stories. They are able to use the computer to verify correct answers for math work done in class.

Grade 2

Children in grade 2 have mastered most of the skills presented to the younger children. They have also explored the STEP function in order to count by 2's, 5's, etc. Most children are able to identify the major parts of the computer (input, output, memory, arithmetic-logic, and control). Some children are beginning to develop skills with the graphic capabilities of the Atari. They can use the PLOT and DRAWTO commands and the coordinate graphing system. The seven year olds are able to type in four or five program lines and many are able to identify the meanings of each line. They are also able to change a program from, for instance, an addition program to a subtraction program. Most second graders can write simple loop programs from memory and some have a basic understanding of INPUT and are able to ask questions and accept answers using this instruction.

Grade 3

Eight year old children are very comfortable with the computers. Their teachers encourage them to experiment in the classrooms with the programs on which they are working. These children are more able to follow the logical thinking processes involved in computer programming and their work is done more quickly with fewer mistakes. Some children are able to write simple programs with PRINT and GOTO from memory. They are able to detect errors and they can identify parts of line statements. Third grade students have a basic understanding of the development of the computer from vacuum tubes, to transistors, to integrated circuits. They can state that computers have gotten smaller and more powerful. These children are better able to read and follow directions during CAI and they have used a greater variety of software.

Grade 4

Children in the fourth grade are able to explore the computer according to their own interests. Some spend much additional time with the computer, others seem to ignore the device entirely. In addition to those concepts learned by the younger children, these students have been introduced to the use of the computer for repeated problem solving through the writing of simple input programs. This skill grew from the concept of area and perimeter which was reinforced in the computer class in order to include it in an understandable program. As these students are beginning the transition from the Piagetian stage of concrete operations to that of formal operations they are also becoming more adept at following the logic behind computer programming. We have noticed that some children are genuinely beginning to internalize programming processes at this level.

Grade 5

At this level children are using additional commands such as CLS and CHR$ (23) for the TRS-80s. They add these into the skeleton programs we develop. They also have explored time loops, IF-THEN with multiple responses, and the use of blank lines between PRINT statements. Moreland has fifth graders with a wide range of interests and intelligence levels and the work at this grade level also evidences this wide range. We have seen more transfer of learning and more creative thinking beginning with this grade level. CAI takes the form of "French Wordsearch" and "States and Capitals" as well as basic math. Students are increasingly interested in simulations and more complex games beginning at this age.

Grade 6

Our oldest students are able to adapt and interpret line statements. They can identify important parts of programs and many of them are excellent editors. Several spontaneously alter programs to meet their own ideas. They have learned well, in most cases, all the commands and instructions taught in earlier grades. The great difference in performance lies in their excellent skills and in their ability to think on an abstract level. They use timer loops, input statements, simple graphics, and other embellishments within their programs without any prompting. GOTO, GOSUB, and READ-DATA are also becoming part of their repertoire.

The basic format for teaching, due to the time limitations in the computer lab, has been to develop a simple program of from 3-5 lines for the youngest students and from 7-12 lines for older students. The skeleton program is built upon in following lessons. As students become more proficient,, they begin to put together small pieces of programs to make more complex programs. This procedure has led to great success for all students because it is success oriented. No child is defeated and no child continually feels as if he has failed.

We have noticed that even children with severe learning difficulties, a history of failure, or short attention spans will spend the entire half-hour class in rapt attention. The computer seems to have a hold on every child, at least at this initial stage. The learning disabilities teacher is using the computer in her room extensively with excellent results. Teachers restrict computer use in the classroom to those who have completed school work and this has increased productivity in the classroom.

A lack of good software has hindered our efficient integration of CAI in the school. Because of our major thrust towards programming, however, we have not suffered too badly. We look toward the near future before suitable software is available.

Moreland teachers all have a positive attitude toward computers and many are beginning to take a genuine interest in learning more about them. They spend time at home with their computers and available material. Many intend to take additional coursework.

In conclusions, the Moreland computer program has begun in a positive and exciting manner. Children are learning at a quick pace and, although changes will take place in the coming years, the goals and objectives seem to be appropriate and feasible ones for elementary age children.

A COMAL SYSTEM FOR IRISH SECONDARY SCHOOLS

Kevin Ryan
Department of Computer Science
Trinity College
Dublin 2, Ireland.
Phone: Dublin 772941 Ext. 1765/1785

SUMMARY

The Comal language originated in Denmark in 1975 as a structured dialect of Basic. Since then it has been revised and defined as a language in its own right and it has been the standard language in Danish schools since 1978. The Irish government has undertaken to provide a microcomputer in every secondary school and intends to encourage Comal as a main teaching language. They have requested the Department of Computer Science of Trinity College Dublin to design and implement a Comal system that is user-friendly, portable, and easily extended. This paper describes the extended Comal language and environment. The first versions of the complete system are to be available by July 1982.

1. THE ORIGINS OF COMAL

Comal has been developed as a structured alternative to Basic. The origins of Basic, the Beginner's All-Purpose Symbolic Instruction Code, predate the awareness that the GOTO was dangerous (Dijkstra/1/) and that correct program structure was a prerequisite to correct programs (Dahl et al. /2/). As a result the Basic language had only one 'structured' construct, the FUR loop, but it gained in popularity and use so quickly that by the early 1970's it was the most common language of small computers and novice users. In 1974 Borge Christensen of the State Teachers College, Tonder, Denmark and Benedict Loftskedt of the University of Aarhus, Denmark designed and implemented some Pascal-like extensions to Basic so that Wirth's ideas in Systematic Programming /3/ could be more readily implemented by students. The effect was considerable. According to Christensen 'the students write better programs than they used to in Basic and their programs have become readable' /4/. Like Basic the Comal language proved more popular, and spread more rapidly, than its authors had anticipated. It was quickly adopted as the standard teaching language in Danish schools and has excited interest in a number of other countries. Its success is due in part to lessons learned from Basic, as well as to the novel language features that have been added.

2. THE MERITS OF BASIC

The popularity of Basic as a first language, and its ubiquity in today's microcomputers, cannot be simply attributed to the ignorance of the novice user or the perversity of human nature. Basic has a number of advantages, especially for first time users, that are frequently ignored by the experts of industry or education. Following Bramer /5/ we can identify these advantages.

(a) Interactive use: Dartmouth's Basic was intended to make the first computing course easier and to take advantage of the arrival of timesharing systems. Later versions of Basic are designed to be interactive and even easier to use. As an interpreter Basic provides line by line syntax checking which reinforces learning of syntax rules and reduces the likelihood of having to correct many occurrences of the same error.

(b) Debugging: Most Basic interpreters provide a simple but powerful debugging facility which permits the user to follow and change the flow of control and even to modify variable values during piecemeal execution of the program.

(c) Editing: Again, the Basic approach was simple to understand and therefore simple to use. The program lines are thought of as so many numbered boxes, any one of which can be changed simply by rewriting it.

(d) Operating System: To the user, Basic appears to have no operating system. Facilities to store and recover programs, to start and stop programs, and to move data to and from the screen, are all part of the Basic language. No distinction is made between system commands and program statements. A novice user is not confronted by a multilevel system embodying a different language at each level, with the requirement that before the most trivial program can be run a set of 'magic' passwords must be learned to get through each of the intervening levels.

All of the above points refer to the Basic environment, not to the syntax of the Basic statements. As a language Basic was no better than its predecessor Fortran, indeed in some respects, notably variable naming and use of subroutines, Basic was worse. But its friendly interactive environment more than counterbalanced its syntactical limitations. To the first time user Basic provided an eminently simple conceptual model (Figure 1), with an obvious relationship between display screen, work area, and (later) external files. In the design of an environment for Comal, one of our prime objectives was to maintain the attractive simplicity of the Basic environment, while providing a language that facilitated development of good structured programs.

3. FEATURES OF COMAL-80

Instead of attempting an exhaustive (and exhausting) definition of the Comal language, it seems more appropriate to emphasize those language features that distinguish Comal from most Basics. These facilities, especially those which aid structured programming, add considerably to Comal's attractiveness. (Some complete Comal programs are shown in the Appendix.)

(a) Structured Constructs: Statements are provided to implement selection and iteration, with full control and without use of GOTO's. Selection is effected by two statements, IF and CASE. In the CASE statement, an OTHERWISE section is optional. If it is omitted and none of the CASEs is selected, the entire statement has no effect.

Repetition is implemented in three forms, FOR, WHILE and REPEAT. The FOR statement provides loops controlled by counters (as in Fortran or Basic). The WHILE and REPEAT statements provide conditional loops that differ only in where the termination condition is tested. Implementing an exit from the middle of a loop remains a problem as it cannot be done without uncontrolled GOTOs or code repetition.

Comal recognizes that GOTOs and structured programming are not incompatible (e.g. Knuth /6/) so a GOTO statement is provided. However transfers are to user-specified LABEL statements rather than to meaningless and error prone line numbers.

(b) Automatic Indentation: It is accepted that a consistently indented structured program is easier to read and understand, so Comal incorporates an automatic listing facility so that standard indentation is applied to all the statements mentioned in (a).

(c) Named Procedures with Proper Parameters: Good program segmentation is long recognized as a major aid to program correctness, clarity, and usefulness (e.g. Parnas /7/, Myers /8/), provided the 'segments' are properly protected. Comal has procedures that are invoked by name and that cannot be entered accidentally. Just as GOTOs in Comal refer to labels, so also the confusing and dangerous 'GOSUB line-no.' is replaced by a procedure invocation. Parameters may be passed by reference or by value, facilitating the creation of truly general purpose routines. Procedures may also be called recursively.

(d) Variable Types and Names: Comal does not require declaration of simple variables Their type, which can be numeric or string, is deduced from the variable name, which in turn may be of 'any' length. Variables in Comal programs can have intelligible names which aid readability and later maintenance.

These features were all present in the draft standard language Comal-80 /9/. Extensions since then, arising from the current project, are discussed in Section 6.

4. THE COMAL PROJECT OBJECTIVES

Government policy in Ireland has recently emphasized the importance of computers in business, industry, and education. Several studies (e.g. /10/) have identified the importance of computer literacy to future economic development. A government commitment has now been given to provide one or more microcomputers to every second level school and to introduce a standard examination for students of computer science. Comal has been chosen as a preferred language for teaching, and the Department of Computer Science at Trinity College Dublin has undertaken to design and implement a suitable Comal system.

The Comal system is to consist of an interpreter for an agreed standard Comal and an environment in which Comal programs may be developed. The system is to be portable, standardized, and efficient.

(a) Portability: In view of the competing and evolving range of microcomputers available to schools, it is desirable to design a system that is independent of any particular machine. Future enhancements and the advent of 16-bit microcomputers make this essential, so that schools' software can be carried forward to later generations of microcomputers.

(b) Standardization: Having a standard system comprising language and environment is important for two reasons. Firstly, a standard examination can be set and fairly graded if all students have used substantially the same system. Secondly, exchange of software or the provision of packages from a central source is facilitated by a standard system.

(c) Efficiency: Both machine efficiency and user efficiency must be considered. As many microcomputers will have less than 64K available in RAM, the system must be kept small enough to allow the user to develop substantial programs in, say, 48K. Response time, particularly during program entry, should be acceptable to an experienced user. This implies avoiding slow floppy-disk input/output as far as possible. User efficiency implies such characteristics as robustness, user-friendliness, and extensibility. The system, especially the environment, should be easy for the novice but also productive for the experienced student (or teacher). In particular it should be possible to implement schools' applications systems (e.g. CAL, student records) with relative ease and confidence.

5. THE PROJECT APPROACH

The approach adopted aims to distance the system from any particular hardware, and at the same time to make use of available software tools. Design of the environment and definition of the language are the first two major tasks, and these can take place in parallel. The Comal language definition is to be in a form suitable for automatic parser generation. The intermediate code produced must allow regeneration of the source code, but must take up a minimum of memory space. At the same time the Comal environment must be designed and implemented. With the addition of action routines a machine independent prototype will become available for development and testing. Transfer to a microcomputer will be done by cross compiling and downloading the entire system.

At the time of writing, the major achievements have been to define an improved version of Comal and to design a suitable operating environment. The new Comal is currently being defined in conjunction with B. Christensen in Denmark and may form a new standard Comal-82. Its salient innovations are described in Section 6, and the environment is discussed in Section 7.

6. EXTENSIONS TO COMAL

Although Comal cannot hide its Basic origins, it was decided to omit Basic features that were dangerous, unhelpful, or misleading. To aid structured programming, some statements were added or changed and the Comal procedure was given some control over access to its local variables. Finally, the need to run large applications in a restricted memory area was recognized by providing a novel PACKAGE facility.

(a) Line Numbers: In Basic, line numbers serve two fundamentally different purposes. As an aid to simple line editing, they are one of the major attractions of Basic's working environment, but as labels for jumps or subroutine calls, they are most inappropriate and error prone. Comal therefore regards them as part of the environment, not intrinsically connected to the statements. If desired, a particular set of line numbers can be stored with a program text. Both 'GOTO line no.' and 'GOSUB line no.' are omitted from the language to be replaced respectively by 'GOTO label' and 'proc. name'. The EXEC or CALL statement is not required as it is considered unnecessary and unnatural. It is more reasonable to write 'GET-CLASS-LIST' than to write 'EXEC GET-CLASS-LIST'.

Distancing Comal from line numbers anticipates the widespread use of screen editors which might well make line numbers redundant.

(b) Structured Constructs: An additional iteration construct is added to allow for orderly termination of a conditional loop, whose test is performed elsewhere than at the beginning or end of the loop. The LOOP statement allows conditional ending of the iteration by an 'EXITIF condition' statement. With the addition of this statement, Comal now allows every possible looping structure to be coded naturally, safely, and without GOTOs. Accordingly, not only are GOTOs into structures statements forbidden, but GOTOs out of such statements are not allowed either.

(c) OPEN and CLOSED procedures: To facilitate protection of local data, and in particular to allow the use of prewritten programs without the inadvertent duplication of names, Comal allows a procedure to be CLOSED or OPEN (default). OPEN procedures can access variables in the main program area and their own local variables may also be changed by other procedures. CLOSED procedures have protected local variables that cannot be modified from outside. However, if a CLOSED procedure wishes to make specific variables available to the world outside, it lists them in a SHARE statement.

CLOSED procedures provide additional security as well as aiding software portability (e.g. using PACKAGES).

(d) The PACKAGE facility: This allows users to create, and later use, groups of application procedures that share common protected data areas. The procedures, which may be in Comal or other languages, are available for the user to invoke but the user cannot access them in any other way. A package is 'loaded' by the USE statement and is 'removed' by the DISCARD statement.

Packages have many advantages. Special purpose packages can be acquired or written for applications such as graphics, CAL, student records, and timetabling. Because of the closed nature of packages there will be no conflicts of variable names between packages, or between a package and the user's program. In this way Comal facilities can be extended indefinitely by advanced users.

A second major advantage of packages is that they allow users to DISCARD a package when it is no longer needed and to USE another package in the same memory area. Complex, but well controlled overlay structures can be created in this way by experienced users. This appears to be a far superior system to the CHAINing of Basic programs.

7. AN ENVIRONMENT FOR COMAL

Initially the Comal environment will provide a minimal set of system commands tailored for beginners, an approach similar to that of Embley and Nagg /11/. As Comal is an interactive language however, its command set will include some of the normal Comal statements which are inputs for immediate execution. In designing the other (non Comal) commands, the following qualities were sought: Orthogonality, Uniformity, Consistency, and Safety.

(a) Orthogonality: New users acquire confidence in a system if generalizations about commands are 'usually' correct. Conversely, if commands have frequent exceptions users never feel confident enough to extrapolate from their current knowledge. Ideally a command set should be such that all combinations of valid keywords with valid operands result in valid commands.

(b) Simplicity: A command set should contain the smallest possible number of different commands, compatible with keeping each command simple and instructive. In particular command keywords should have meanings similar to those of everyday use. This is more important than finding contrived synonyms to achieve unique abbreviations.

(c) Consistency: The semantic effects cf commands should be predictable to the learner. Defaults should be 'reasonable' and where no reasonable default exists the user should be prompted for further information.

(d) Safety: The novice user can be assumed to make every possible error. The system should therefore be robust and provide good error messages. To protect the user however, the system should also make it difficult to unintentionally make irretrievable decisions.

Software development environments have recently been criticized for their poor user interface (Prentice /12/). In contrast, the Comal environment provides a simple one-level system that has no modes or submodes. However the set of commands available to the user can easily be extended (using PACKAGEs). At present there is a provision for a working subset of commands that cover program execution, editing, filing, and debugging. A special PACKAGE will include commands to create, edit, and save PACKAGEs. Because the environment is on the 'same level' as the workspace in which the user creates programs, the novice programmer sees the commands as part of the Comal language. This perception is reinforced by the availability of some Comal statements as commands (e.g. assignment, PRINT). As this unified world view has been one of the major attractions of Basic it is important that Comal maintains this approach.

The Comal environment can be illustrated by a selection of its commands.

(a) The LIST Command: The user's perception of the Comal system is shown in Figure 2. The operation of LISTing is taken to be any transfer of data that results in a visible RESULT. This allows many possible LIST verbs, described schematically as follows:

```
   LIST  [source]  [ON target]
                                ┌WORKSPACE
   where source is one of      {  filename
                                └SCREEN

   and target one of   ┌SCREEN
                       └PRINTER
```

The defaults are underlined. The operation 'LIST' causes the workspace to be listed on the screen.

(b) The COPY Command: In contrast to LIST, the COPY command produces output that is not visible to the user. Its format is:

```
   COPY [FROM {WORKSPACE}][TO {filename }]
        [     {filename }][   {WORKSPACE}]
```

This command provides facilities equivalent to Basic's SAVE and LOAD, as well as file COPYing. The words SAVE and LOAD are frequently misleading to newcomers, implying as they do the loss of the source text from memory. 'COPY' is a more accurate description of what actually happens.

(c) The DELETE Command: As this command does destroy a specified source, it does not take effect without a positive user response to a challenge. The command format is:

```
   DELETE      /filename/
```

If 'filename' is omitted the delete refers to the workspace but only after confirmation by the user.

(d) The DISCARD Command: This allows the user to remove packages that are no longer needed. Its format is:

```
   DISCARD      /package-name/
```

If package-name is present, then that package's procedures are no longer available for use. If no package-name is given, the default is for all packages to be removed but only after a positive user response to a challenge.

8. CONCLUSIONS

The project has further developed a structured programming language which is suitable for novice programmers but will provide sufficiently comprehensive facilities for advanced applications. The concept of a designed environment tailored to novice use has been combined with the Comal language to give a system design that is simple, powerful, and user-friendly. The method of implementation using a systems programming language and a machine independent intermediate code, ensures the widest possible portability of the system. Sizing and timing information, as well as practical user experience, should become available when the prototype system is delivered in July 1982.

ACKNOWLEDGEMENTS

The work described in this paper is being carried out by the Comal group of the Department of Computer Science, Trinity College Dublin. The group members are:

J.G. Byrne
K.T. Ryan
C.R. Kitchen
M. Mac an Airchinnigh
B. Lynch
R. Ryan

Their assistance in preparing this paper is gratefully acknowledged.

REFERENCES

1. Dijkstra E.W. "GOTO Statement considered harmful" CADM 11(3) pp. 147-148 1968.
2. Dahl D.J., Dijkstra E.W., Hoare C.A.R. Structured Programming Academic Press 1972.
3. Wirth N. Systematic Programming: an Introduction Prentice Hall 1973.
4. Christensen B. "Programming Languages for Beginners: the Global Challenge" Computer Bulletin p. 20 Sept. 1981.
5. Bramer M.A. Comal 80 - Adding Structure to Basic CAL Research Group Technical Report No. 3 Open University England 1980.
6. Knuth D.E. "Structured Programming with GOTO Statements" Comp. Surveys 6(4) pp. 261-301 1974.
7. Parnas D.L. "On the Criteria to be Used in Decomposing Systems into Modules" CACM 15(2) pp. 1053-1058 1972.
8. Myers G.L. Reliable Software Through Composite Design Petrocelli/Charter 1975.
9. Comal-80 Working Group. Comal-80 Standard Denmark 1980.
10. NBST Microelectronics - the implications for Ireland - Preliminary Report National Board for Science & Technology Ireland 1980.
11. Embley D. & Nagg G. SIMPLE - A Programming Environment for Beginners SIGCSE Bulletin 13(4) pp. 7-12 1981.
12. Prentice C. An Analysis of Software Engineering Environments Soft. Eng. Notes 6(5) pp. 19-27 1981.

APPENDIX PROGRAM 1

```
// COMAL
// PRINT THREE NUMBERS IN SORTED ORDER
//
INPUT "A=" : A, "B=" : B, "C=" : C
IF A > B THEN
   IF B > C THEN
      PRINT "A,B,C"
   ELIF A > C THEN
      PRINT "A,C,B"
   ELSE
      PRINT "C,A,B"
   ENDIF
ELIF A > C THEN
   PRINT "B,A,C"
ELIF B > C THEN
   PRINT "B,C,A"
ELSE
   PRINT "C,B,A"
ENDIF
END
```

```
// COMAL   PROGRAM 2
// MAIN ROUTINE FOR CAI SESSION
// USES PACKAGES SOME OF WHICH TAKE ALL
// AVAILABLE RAM.
//
WELCOMEMESSAGE             // ON SCREEN
REPEAT
   MAINMENU
   INPUT "CHOSEN COURSE CODE (S TO STOP)" :
                                          OPTION$
   CASE OPTION$ OF
   WHEN 'G'
      PRINT "GEOGRAPHY"
      USE GEOGPACK
      GEOGMENU
   WHEN 'M'
      PRINT "MATHEMATICS"
      USE MATHPACK
      MATHMENU

   {ETC.}
   WHEN 'S'
      PRINT "STOP"
   OTHERWISE
      PRINT "NO SUCH COURSE : RE-ENTER"
   ENDCASE
UNTIL OPTION$ = "S"
END
```

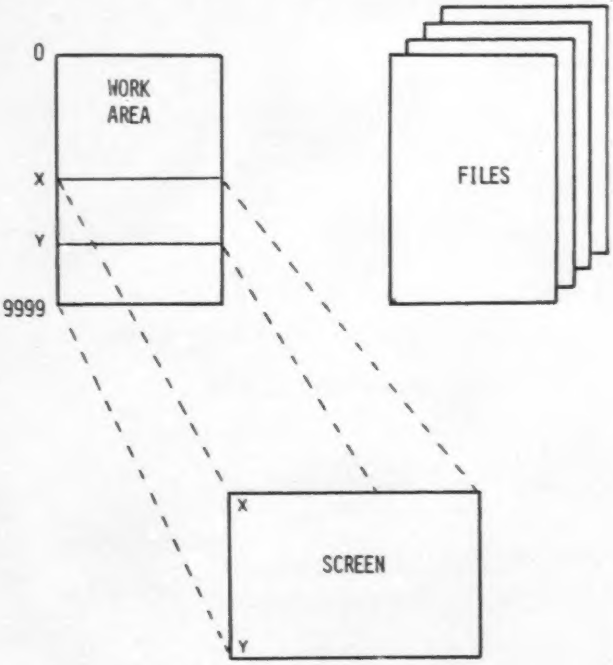

0
WORK AREA
X
Y
9999
FILES
X
SCREEN
Y

Figure 1 BASIC's Conceptual Model

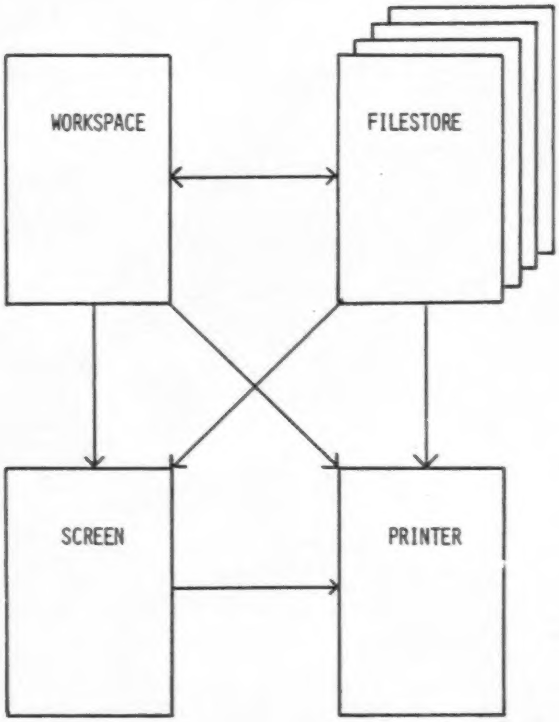

WORKSPACE
FILESTORE
SCREEN
PRINTER

Figure 2 COMAL User's Model

Special Session

FACULTY TRAINING AND DEVELOPMENT IN IN ACADEMIC COMPUTING

Chaired by: Sheldon P. Gordon
Suffolk County Community College

ABSTRACT

The focus of this session will be on the types of problems which arrive for project directors of large scale projects in academic computing and the strategies that have been successful in overcoming them. A report will be given on the special meetings of project directors of computer related projects under CAUSE held by the NSF last year in Norfolk, Virginia, concurrent with NECC 81. Experiences of project directors will be emphasized which might be helpful to other people planning to implement similar projects.

Special Session

PANEL ON PROGRAM PROJECT MANAGEMENT

Chaired by: Rita Virginia Rodriquez
Department of Mathematics
University of Puerto Rico
Rio Piedras, Puerto Rico 00931

ABSTRACT

Much federal and other funding has been and is being given for the execution of software development projects at institutions of higher education. Many of the projects so sponsored are directed and carried out by professors with minimal previous experience in managing a programming project and who often have little knowledge of the capabilities of computers and programmers. This panel discusses several aspects of program project management, including some of the standard procedures and tools of the trade. Special consideration will be given to the peculiar problems of carrying out such projects in the university environment: funding, student programmers, accountability for results, getting along with The Administration, and other areas. The panel will also take a critical look at the results of poor management. The panelists all have considerable working experience in a wide variety of programming projects.

Participants

Herman A. Acuna
Faculty of Natural Sciences
University of Puerto Rico

Rio Piedras, Puerto Rico 00931

Frank D. Anger
Department of Mathematics
University of Puerto Rico
Rio Piedras, Puerto Rico 00931

Jose Babilonia
Faculty of Business Administration
University of Puerto Rico
Rio Piedras, Puerto Rico 00931

Hiram Cabaza
Faculty of Engineering
University of Puerto Rico
Mayaguez, Puerto Rico 00708

398

Project Presentation

THE COMPUTER IN THE UNDERGRADUATE MATHEMATICS CURRICULUM

Chaired by: Mike Mulder
ESI, Inc.

Nicholas Arganski
Noel Evans
Robert Evans
Joyce Friske
Dale T. Hoffman
John Kontogaines
John C. Miller
Preston Prigger
Dale A. Schoenefeld
Carla J. Thompson

ABSTRACT

Calculus Tutorial System

Nicholas Arganski, Director College D'Enseignement General Et Professionnel, P.O. Box/CP 2000, Ste. Anne de Bellevue, Quebec H9XS19

The project originated at John Abbott College, Ste. Anne de Bellevue (Montreal) Quebec, and constitutes a complete course in Calculus, presented in an unique and innovative way.

The CTS consists of five main subsystems: The Interactive Derivative Solving System (IDSS), the Interactive Integral Solving System (IISS), the John Abbott College Individual Supplementary Calculus Instruction (JACISCI), the Computer Generated Individual Problem Sets (CGIPS) and the GRAFIT - a practical graphing program. Of these, the IDSS, the IISS, the NEWPROB of the JACISCI and the CGIPS constitute unique and innovative programs which originated at John Abbott College.

The IDSS and the IISS enable a first year student of Calculus to find derivatives and integrals (with explanations) of any algebraic or transcendental function that he/she might come across.

The NEWPROB of the JACISCI enables any non-programming teacher to prepare a lesson (a series of problem files) on any topic of this choice in an easy and effective manner.

The JACISCI processes students' on-line lessons prepared with the help of the CAGES (Calculus Assignment Generating and Evaluating System) series of programs. It provides immediate feedback and grading of students and keeps extensive records.

Finally, the CGIPS offers five different approaches to computer testing with extensive records and statistical results.

All phases of CTS are being currently re-evaluated and improved.

ABSTRACT

Graphic Explorations

Robert Evans, Mathematics Dept., Cuyamaca College, El Cajon, Calif. 92020

A series of modules (programs) have been developed which allow the student to easily explore the relationships between the algebraic and graphic forms of all the functions and relations usually considered in precalculus mathematics including:

Cartesian coordinate system
Polynomials
Conic Sections
Logarithmic
Polar coordinate system
Linear equations
Rational functions
Exponential
Trig functions

Each module has three operational modes:

EXPLORATION - Allows the student complete control of all functional parameters through easily operated menus or commands. For example, in the linear module the student has control of the slope and intercepts. Once the parameters are selected and values provided, an appropriately scaled high resolution graph is produced. Graphs can be overlayed to study the effect of a particular parameter. No computer or programming knowledge is assumed or required.

GAME - Automatically sets the parameters within the function being studied. An appropriately scaled graph is presented to the student. The student is then presented a question concerning the graph and the value of one of the functional parameters. For example, if a straight line were displayed, one possible question would be: What is the slope? Responces are timed and a score is assigned based on accuracy and speed. A fixed number of problems constitute a game.

QUIZ - Similiar to the game mode. However, in this mode the student is not timed and student responces are logged for later scoring and analysis by another program available only to the faculty member.

The use of a "discovery" approach has been received enthusiastically by the students. Student insight into the relationships between functional parameters and graphical form appears more complete. Finally, the use of the computer and its graphic capability has reduced the time for students to develop the necessary insights and proficiencies related to translating from one form of a function to another.

ABSTRACT

Computer Aided Education in Calculus -- Limits

Dale Hoffman, Dov. of Science and Mathematics, College of the Virgin Islands, St. Thomas, Virgin Islands 00801

Many of the fundamental concepts of calculus are inherently graphical and are well-suited for demonstration on microcomputers with graphics capabilities. The programs described in this presentation are an attempt to treat the single topic of limits in some detail and from several perspectives. The programs have been used by students in introductory calculus classes and are primarily intended for individual student use. They are written in Applesoft Basic for use on an Apple II Plus (48K). A color minitor is recommended but is not essential. The programs include:

1. FINDING LIMITS is a utility program for graphically determining limits. It uses the idea of repeatedly magnifying the graph of the function near the limit in order to study its behavior there.
2. EPSILONS & DELTAS is a program to help students understand the definition of a limit. The graph of a function and a neighborhood of the limit are shown, and the user is asked to graphically determine which satisfies the definition. The program and graphics are very interactive.
3. LIMIT PROBLEMS randomly generates several types of limit problems for student practice. The type and level of difficulty are user-selected. Hints and solutions are given.
4. GRAPHIC PROBLEMS generates a "piecewise random" graph. It then generates questions, referring to the graph, about one and two-sided limits, continuity at points, and continuity on intervals.

The integration of computer use into science and mathematics courses at the College of the Virgin Islands is supported by a MISIP grant from the National Science Foundation. This particular project is supported by a grant from the Foundation for the Advancement of Computer-aided Education. These programs will eventually be available from the Foundation.

ABSTRACT

FUNGRAPH: A Language for Computer Inspired Learning

John C. Miller, The City of the CUNY, New York, NY 10031

FUNGRAPH is a programming language designed to inspire students to explore mathematical concepts and programming techniques simultaneously. It allows the user to experiment with mathematical ideas in the course of producing visually interesting output on the computer screen. He does this by writing programs using simple commands and rules of syntax, selected to provide practice in manipulating essential mathematical objects according to standard mathematical conventions.

The FUNGRAPH language forms a natural bridge from the user's conceptualizations to the finished output on the computer screen. It is a structured language (in the technical sense that the term is used in computer science), and the concept of subroutine is introduced in the first or second session through a natural process of defining new commands to supplement the initial FUNGRAPH vocabulary. A subroutine call is then accomplished by using the new command in subsequent programs. (Experienced programmers will recognize the concept of "word" as used in the language FORTH.) A natural method of repeating a process is then the use of a recursive subroutine call with a local parameter as a counter. By contrast, only a comparatively advanced BASIC programmer can hope to use the BASIC GOSUB command recursively, and even the notion of subroutine does not occur naturally to beginning BASIC programmers.

FUNGRAPH uses conventional Cartesian co-ordinates as the basis for its output, rather than the artificial TAB or SET commands (with inverted y-axis!) utilized by most versions of BASIC. Translating, dilating and compressing co-ordinate axes are the natural means for a beginning FUNGRAPH user to obtain the visual output that he fancies.

For the more advanced uers, FUNGRAPH provides extensive and sophisticated function-graphing capabilities. The concepts of domain and function are quickly assimilated through the FUNGRAPH commands DOMAIN and FUNCTION. Composite functions are encountered naturally; the values of both he intermediate and final functions are alculated and displayed and graphs drawn eadily via simple commands. The easy interchange of co-ordinate axes provides a atural way to discover and master the principle of the inverse of a function. The user can also readily plot families of urves by varying parameters, and plot urves, or families of curves, using parametric representations.

The project presentation will introduce the main features of the FUNGRAPH language via presentation of a series of sample programs of increasing complexity, including a number written by students. Emphasis will be on the pedagogical potential of the language as a new and attractive means of inspiring exploration of mathematical ideas and concepts.

ABSTRACT

An Undergraduate Curriculum in Computer Graphics

Preston Prigger, Maycrest College, 1607
W. 12th Street, Davenport, Iowa 52804

We have developed a unique approach to the study of Computer Graphics. We have combined the visual expression skills and techniques of traditional Art courses with the technology available in Computer Science.

This approach is possible through a co-operative major offered jointly by the Art and Computer Science departments. The student is exposed to concepts of line, shape, color and light in coursework from Art. Dimensional representation and perspective are included from courses in sculpture and painting.

The Computer Science component exposes the student to coursework in algorithms, data structures and the programming language necessary to impliment the concepts learned in Art on the computer. The focus of the Computer Science component is on the development of interactive computer graphics.

Both departments are "project" oriented and this encourages the development of a necessary experiential base. Two courses are central to the curriculum: Principles of Interactive Graphics and Seminar in Comuter Graphics. These courses explicitly deal with integrating the conceptual bases of both disciplines. The Seminar is team taught by one person from the Art department and one person from the Computer Science department.

We are involved in a project to develop new means of graphic information presentation with our television studio and our students have the opportunity to work in CAD/CAM through a local manufacturing company.

ABSTRACT

Laboratory Intergration in Finite Mathematics and Survey Calculus

Dale A. Schoenefeld, PhD, Div. of Mathematical Sciences, University of Tulsa, 600 S. College, Tulsa, OK 74104

This presentation describes the development, implementation and evaluation of computer based materials for a one year mathematics sequence taught at The University of Tulsa to students in business, nursing, and the life and social sciences. Topics in the sequence include elementary algebra, matrix theory, linear programming, mathematics of finance, probability, statistics, differential calculus, and integral calculus.

The computer-based materials include many that originated and were developed locally and others that are available from other sources. Some of the materials have been collected into a student laboratory library that requires active student involvement with the university time sharing computer. Other materials have been collected as an instructor's classroom demonstration library. All materials available for student utilization are in ANSI 77 FORTRAN, generally subset. Most materials available in the instructor's demonstration package are implemented on an Apple II computer using ANSI 77 subset FORTRAN with turtle graphics.

The project is nearing the conclusion of support by the National Science Foundation under the Local Course Improvement program. The materials are not of the drill and practice variety and students are not expected to be computer programmers. Rather, the primary objective of the project is to motivate and reinforce basic concepts normally included in the indicated course sequence. An important secondary objective is to improve the level of computer literacy among the target audience and, in particular, to expose them to the computational techniques made available by computer technology.

ABSTRACT

Microcomputers in a Two-College Center for Learning and Instruction in Mathematics - Concepts Building

Dr. Noel Evans, Mathematics Instructor, Tulsa Junior College, Tulsa, OK 74115;
Mrs. Joyce Friske, Mathematics Instructor,
Dr. John Kontogaines, Director, Math, Science and Engineering; Dr. Carla Thompson, Mathematics Instructor

Description:

This presentation will focus on the acquisition and use of microcomputers in a two-year college C.L.I.M.B. Center (Center for Learning and Instruction in Mathematics - concept Building). The presentation will address specific concerns associated with integrating microcomputers into a predominantly main frame junior college setting. Problems associated with integrating microcomputers into a non-computer science department will be examined.

A panel of three instructors and one administrator will discuss various aspects of developing a microcomputer-oriented mathematics center. The panel will describe the major considerations and activities that were encountered in the pre-operational year and first-operational year of the C.L.I.M.B. Center. Specific areas of discussion include:

1. Rationale for Microcomputer in C.L.I.M.B. Center
2. Micro vs Mini vs Mainframe
3. Selection, Purchase, and Acquisition
4. Use and Evaluation

The panel will provide audience participants with specific guideline lists for each of the four areas discussed. A brief question-answer period will conclude the session.

Institutional Profile:

Tulso Junior College is the third largest institution of higher education in the state of Oklahoma. Supporting two full time campuses and three part time campuses, Tulsa Junior College has an enrollment of 13,751 including over 50 vocational-technical programs. The C.L.I.M.B. Center (Center for Learning and instruction in Mathematics-concept Building) is located at the Northeast Campus and is open to all students.

Author Index

Acuna, Herman 380
Adams, James M. Jr. 346
Adams, Mark 231
Adams, Ron 264
Alvir, Howard P. 40
Anderson, Ron 38
Anger, Frank D. 380
Arganski, Nicholas 381
Arrieche, Petra Lucia 313
Austing, Richard 311
Avery, Robin D. 229

Babylonia, Dr. 380
Baird, W.E. 331
Baker, John M. 351
Basham, Charles W. 76
Baxter, Robert 133
Beidler, John 165
Belton, Sandra Y. 268
Beougher, Elton 230
Berghel, Hal 263
Billings, Karen 169, 312
Booker, Don M.M. 76
Bork, Alfred 11, 43
Borse, G.C. 196
Bowyer, Kevin 364
Brandt, Richard C. 107
Bronson, Richard 74
Brown, Warren R. 166
Brubaker, Thomas A. 105
Bruckhart, Glenn 323
Bryant, Dudley F. 37
Bulinski, Richard L. 28
Burns, George 268
Butler, Matilda 313

Caccano, James M. 272
Cain, Tom 105
Carr, Douglas 314
Carver, Jan 37
Chaffin, Jerry 269
Clark, A. Rees 78
Clonninger, Todd 231
Clump, C.W. 196
Collins, Donald 231
Collins, Victoria 231
Coltharp, Forrest L. 230
Conklin, George 289
Conrad, Lora P. 59
Cook, Michael 268
Cooper, Martha 40
Corbitt, Mary K. 230
Cornelius, Richard 95, 231
Culp, George 264, 312

Dahl, Thomas 65
Dale, Nell 263
Deal, Ralph 74, 110, 129
Deimel, Leonel E. Jr 114
Denenberg, Stewart A. 81
Derbyshire, Elizabeth N. 76
Dillard, E. Margo 141
Dillingham, Larry 348
Dooley, Tom 97
Driver, Richard T. 230
Dubitsky, Barbara 269
Duncan, Karen A. 311, 345

East, Philip 347
Ellis, Margaret 110
Etlinger, Henry 263
Evans, Amanda L. 1
Evans, Noel 383
Evans, Robert 381

Fey, James T. 167
Fink, Pamela 203
Finkel, LeRoy 38, 279
Fletcher, Lincoln 318
Flynn, Thomas 77
Foster, Susan J. 78
Franklin, Stanley P. 120
Franklin, Stephen 263
Franz, John M. 231
Friesen, Charles D. 168
Friske, Joyce 383

Gagnard, Alice 111
Garson, James W. 107
Geist, Robert 203
Geens, Dirk L.C. 271
Goodson, Bobby 235, 346
Gordon, Sheldon P. 75, 379
Gould, Thomas M. 38
Goulet, Daniel V. 165
Gowdy, John 105
Grady, Charles 168
Greenberg, Melanie Wolf 79
Greenwell, Raymond N. 75
Grobe, Sue 265
Gruska, Josef 335

Hacker, Rowena 314
Hale, Mark P. 75
Heller, Paul S. 234
Heller, Rachelle S. 39
Hertlein, Grace C. 238
Hickerson, N.L. 24
Himmelblau, David M. 1
Ho, Thomas 165
Hoffman, Dale T. 382
Hogan, John T. 234
Holmes, Glyn 48
Hopper, Judith A. 323
Horn, Carin E. 108
Hoyle, Lawrence 6
Humes, Craig 168

Inagaki, Toshiyuki 1

Jackson, Thames F. Jr. 155
Jensen, Carl B. 54
Jerauld, Lloyd A. 167
Johnson, Dale M. 111

Johnson, Jerry 113

Katz, Sidney 345
Kay, Alan 106
Keller, Sister Mary K. 37
Kelly, Jim 167
Killam, Rosemary N. 141
King, Dennis R. 306
Knapp, Barbara H. 107
Kontogaines, John 383
Kosel, Marjorie 38
Kosterman, Don 213
Kraft, Robert 168
Kraines, David 203
Krause, Marcia 269
Kroeber, Donald W. 15
Krutz, Ronald 105
Kushan, Sister Barbara 269

Laxer, Cary 364
Lazerick, Beth 369
Lee, Gail S. 76
Lemos, Ronald S. 170
Lewin, Ann W. 106
Lewis, H. Rylan 186
Luehrman, Arthur 106, 312
Lunning, Cynthia A. 314

McInerney, Michael F. 225
McKee, Timoghy C. 155
McNulty, James W. 248
Martin, C. Dianne 39
Mathis, Robert 317
Mead, Graham P. 24
Miller, John C. 382
Min, K.S. 213
Mins, Ted 347
Mitchell, Maurice 314
Moberg, Thomas 37, 65
Moeller, Arthur C. 138
Moffat, David V. 114
Monk, John 245
Moore, Margaret L. 254
Moore, William R. 133
Motteler, Zane C. 209
Moursund, Dave 346

Neil, Ruth M. 265
Neufeld, Helen H. 232
Niccolai, Marino J. 186
Niederjohn, Russell J. 138
Nickels, Herb 264

Paisley, William 78
Palmer, John L. 248
Pangora, Paul 106
Parkhurst, Anne M. 302
Pogue, Richard E. 311, 345
Poirot, Jim 312
Polovney, Joseph L. 84
Pratcher, Sandy 312
Prigger, Preston 383
Prigger, Sandra 41
Pulat, Barbur Mustaf 258
Purcell, Edward 271

Ragan, Tillman J. 348
Ray, Mel 364
Reber, Elaine 111
Rennert, Robert A. 37
Rezac, John P. 230
Rhoads, Rosalie 348
Ricketts, Dick 346
Rice, Billie Ann 75, 263
Rinehart, Frank 108
Riva, Eduardo 74
Rivers, Robert 326
Rodriguez, Rita V. 380
Roberts, Nancy 74, 349
Rockhill, Theron 232
Rosenthal, Lynn 108
Ruggles, Marilyn 269
Ryan, Kevin 373

Schilmoeller, Gary L. 306
Schilmoeller, Kathryn L. 306
Schmiedekamp, Carl 216
Schnippert, William 315
Schoenenfeld, Dale A. 111, 383
Schwartz, Helen J. 108
Scott, Jerry 192
Seeff, Adele 263
Shelly, Maynard W. 6
Shirley, Bill 341
Shotwell, Stevan R. 355
Shub, Charles M. 74, 123
Sjoerdsma, Ted 285
Smith, Larry K. 346
Southwell, Michael G. 41
Spain, James D. 91, 233
Spuck, Dennis W. 346
Steen, Lynn Arthur 69
Stephens, Jerry W. 101
Stone, Antonia 169
Stone, Sara 111
Sullivan, Blair 106
Sunkel, Mary Jane 40

Taffe, William J. 359
Tharp, John R. 266
Thomas, Barbara 266
Thompson, Carla J. 383
Trowbridge, David 11
Truax, Randy 97

Underhill, Robert 349

Verkade, Mary 65
Vinsonhaler, John F. 161
Vockell, Edward L. 326

Wagers, William D. 108
Wang, Chaing 258
Ward, Susan 111

Watkins, Robert W. 272
Watson, James 112
Watson, Nancy 112
Weissman, Stephen M. 359
Whipple, Alisha 168
Whitacre, Tamara 6
Williams, Kenneth E. 225
Wilson, Kara Gae 169
Wolverton, William D. 161

Yoshida, Glenn 42

Zamora, Ramon 106
Zglicznynski, Susan M. 315

Subject Index

This index lists by topic papers published in the proceedings of NECC from 1979 thru 1982

ADMINISTRATION

Computer Assisted Academic Departmental Scheduling
    Prather and Rabia, 1981, P. 39
A Financial Planning Model
    Horton and Eerkes, 1981, P. 45
A User-Oriented Classroom Use Program for Colleges and Universities
    Luce, 1981, P. 49

APPLICATIONS AT THE HIGH SCHOOL LEVEL

Project C.A.I.S.H.: Computer Assisted Instruction and Support for the Handicapped
    Brown, 1982, p. 166
Microelectronic Technology: Implications for Secondary School Mathematics Curricula
    Fey, 1982, P. 162
The TOCS (Teacher Oriented Computer System) Approach to Remedial Mathematics
    Jerauld, 1982, p. 167
Cross Lingual Mathematics Information System
    Kelly, 1982, p. 167
Compu-Grade: Teachers Gradebook Programs
    Kraft and Friesen, 1982, p. 168
Students Teaching Students About Computers
    Whipple, Grady, and Humes, 1982, p. 168
The Implementation and Evaluation of Microcomputer Use by Incarcerated Juveniles at the Spofford Juvenile Center
    Stone and Billings, 1982, p. 169
Precollege Software Evaluation and Administration
    Billings, Watt, Mins, and Boody, 1981, p. 279
Report of Activities of the ACM Elementary and Secondary Schools Subcommittee
    East and Rogers, 1981, p. 242
The Teachers Desk of Tomorrow Today
    Wilson, 1982, p. 169

APPLICATIONS IN THE UNDERGRADUATE COLLEGE CURRICULUM

Barriers to the Use of Instructional Computing in the Teaching of Undergraduate Chemistry
    Nuefeld, 1982, p. 232
Computerized Data Logging
    Adams, Clonninger, and Collins, 1982, p. 231
Microcomputers and Large Screen Projection in Science Lecture Halls
    Cornelius, 1982, p. 231
Simulation: Real World Applications and Their Implications for Undergraduate Course Content
    Roberts, Bronson, Deal, Riva, and Shub, 1982, p. 74
The SUMIT Courseware Development Project
    Spain, 1982, p. 233
SUNY Brockport Science and Mathematics Microcomputer Project
    Rickhill, 1982, p. 232

BASIC SKILLS IMPROVEMENT

CAI for Developmental Writing: Going Beyond the Fill-In Exercise
    Southwell, 1982, p. 41
Computer-Assisted Instruction in LD Math
    Prigger, 1982, p. 41
Correcting Deficiencies in Science Backgrounds
    Yoshida, 1982, p. 42
Keyboarding, Touch Typewriting
    Cooper and Sunkel, 1982, p. 40
Using CAI to Teach Adults as a Prelude to Vocational Job Preparation Courses
    Alvir and Carr, 1982, p. 40

BUSINESS/ECONOMICS

BIZNES: A Cuccessful Approach to Simulation in Economic Education
    Schenk, 1981, p. 4
Computer-Assisted Instruction for Large Classes in Economics
    Paden and Barr, 1979, p. 227
Computer Science and MIS College Students: An Investigation of Career Related Characteristics
    Jordan, 1980, p. 7
Computer Simulated Management Training Better than Case Study
    Burke and Callahan, 1981, p. 89
A Computer Simulation Model for Teaching Intermediate Macroeconomics
    Silvia, 1981, p. 1
Force Feeding SPSS in Market Research and Analysis at Hampton Institute
    Wehrle, 1980, p. 12
Short-Run Forecasting of the U.S. Economy
    Bowman, 1980, p. 16
Simulating the Great Depression in Introductory Macroeconomics
    Schenk, 1979, p. 179
A Small Instructional Macroeconomic Model of the United States Economy
    Werner, 1979, p. 189
Students Confront Data Base
    Gottschalk and Elliott, 1981, p. 83
Psyciology of Small Data Processing Systems
    Vazsonyi, 1981, p. 86

THE COMPUTER AS A TOOL IN THE UNDERGRADUATE COLLEGE CURRICULUM AND BEYOND

Academic Computing in Small Colleges: Problems and Solutions
    Keller, Bryant, Carver, Moberg and Rennert, 1982, p. 37
Computer-Assisted Instruction Project
    Ellis, 1982, p. 110
A Computer Information, Analysis, and Simulation System for Teaching in Agriculture
    Basham, Lee and Derbyshire, 1982, p. 76
Development of a Computer Assisted Tutorial in the Use of SPSS for Application to Health Science Agencies
    Greenberg, 1982, p. 79
Integrating Data Flow Models and System Dynamics in Information Systems Design Courses
    Booker, 1982, p. 76
Learning How to Bank at Home
    Gagnard, 1982, p. 111
Microcomputer Facility
    Ward and Reber, 1982, p. 111
Modeling Chemical Reactions in PASCAL
    Deal, 1982, p. 110
Social Research Simulation on a Microcomputer
    Paisley, 1982, p. 78
Using a Microcomputer Network for Instruction of all ages
    Watson and Watson, 1982, p. 112
TEST: Test Evaluation and Score Tabulation
    Schoenefeld and Johnson, 1982, p. 111
Use of Videotape to Present an Introduction to Computing
    Foster and Clark, 1982, p. 78
Word Processing's Usefulness for Writing Instructions
    Flynn, 1982, p. 77

COMPUTER ASSISTED INSTRUCTION/COMPUTER MANAGED INSTRUCTION

College CAI Projects in the Basic Skills Areas
    Wood, Bass, Ries, Heller, Eveland, and Southwell, 1981, p. 37
College CAI in Content Areas
    Bertsche, Spain, Parrish, Arenson and Wozniak, 1981, p. 113

Computer-Based Education: Research, Theory, and Development
Hall, 1979, p. 7
Computer-Managed Education: Providing Quality Education on a Large Scale
McGlinn and Carmony, 1979, p. 44
Computer Based Relativistic Learning
Shelly, 1982, p. 6
Determination of Informational Paths Among Educational Modules
Evans, Inagaki and Himmelblau, 1982, p. 1
The Enhanced CAI Tutorial
Jensen, 1982, p. 54
Groups and Computer Based Learning Materials
Trowbridge and Bork, 1982, p. 11
Ingredients for Excellence in Computer-Based Education Systems
Bailey, 1979, p. 2
Integrating CAI into the Curriculum
Holmes and Kidd, 1982, p. 48
Interaction in Learning
Bork, 1982, p. 43
The Non-Technical Factors in the Development of CAI
Mocciola, 1980, p. 236
An On-Line Directory of CAI Resources
Swigger and Maner, 1979, p. 48
Precollege CAI
Radspinner, Dalton, Smith, Scott, Slaton, Helwick, Smith, Brooks and Beltran, 1981, p. 193
The State of the Art of Instructional Computing: A Report from CONDUIT
Johnson, 1979, p. 14
A Successful Computerized Management System for Individualized Instruction
Stoaks, 1979, p. 52
Towards the Development of a Computer-Managed Instructional System in Pharmacy, Based Upon Integrated Database System Concepts
Henkel and Williamson, 1979, p. 360

COMPUTER AWARENESS PROJECTS

A Computer Elective Course for Seventh and Eighth Graders
Hacker, 1982, p. 314
Computer Literacy in a Small Liberal Arts College
Schnippert, 1982, p. 315
Computers for Popular Consumption
Lewin, Kay, Juehrmann, Pangora, and Zamora, 1982, p. 106
Electronic Information Officer
Butler, 1982, p. 313
Impact of Copyright Laws on Computers in Education
Anderson, Finkel, Gould and Kosel, 1982, p. 38
Problem Instruction for Microcomputers
Arrieche, 1982, p. 313
SDIP Activity III
Mitchell, Carr, and Lunning, 1982, p. 314
Your Future with Computers: A Career Discovery Workshop for High School Students and Their Parents
Zglicznynski, 1982, p. 315

COMPUTER BASED EDUCATION

BAFFLES: A New Deductive Game
Spain, 1982, p. 91
A Call for the Study of Computer Games
Denenberg, 1982, p. 81
Computer-Based Education: Strategies for Success
Fauley, 1981, p. 278
Computer-Based Relativistic Learning
Shelley, Hoyle, and Whitacre, 1982, p. 6

Delivery of College Courses Using Educational Technology
Polovney, 1982, p. 84
Determination of Information Paths Among Educational Models
Evans, Inagaki and Himmelblau, 1982, p. 1
The Enhanced CAI Tutorial
Jensen, 1982, p. 54
Groups and Computer Based Learning Materials
Bork and Trowbridge, 1982, p. 11
Integrating CAI into the Curriculum
Holmes and Kidd, 1982, p. 48
Interaction in Learning
Bork, 1982, p. 43
Language and Music Instruction in the Computer Environment
Raben, Arenson, Fosberg, and Geens, 1982, p. 271
Protecting Interactive Programs for User Input Errors
Maleske, 1981, p. 12

COMPUTING CURRICULA

A Computer Science Major in a Small Liberal Arts College
Mayer, 1980, p. 290
An Educational Program in Medical Computing for Clinicians and Health Scientists
Hybl and Reggia, 1980, p. 276
Implementing Model Curricula
Goulet, Beidler and Ho, 1982, p. 165
The Proper Role of the Computer Science Program by a Small Undergraduate Liberal Arts University
Horton, 1979, p. 197
Required Freshman Computer Education in a Liberal Arts College
Wetmare, 1980, p. 139
A Segmented Structure for Remedial Graduate Computer Science Instruction
Pozefsky and Tharp, 1979, p. 201

COMPUTER GAMES IN INSTRUCTION

Shall We Teach Structured Programming to Children?
LaFrance, 1980, p. 261
Structured Gaming: Play and Work in High School Computer Science
Moshell, Amann, and Baird, 1980, p. 266
Tapping the Appeal of Games in Instruction
McKay, 1980, p. 271

COMPUTER LABORATORIES IN EDUCATION

The Computer Lab of the 80's
Brown, 1980, p. 256
The Education Technology Center
Bork, Franklin, and Kurtz, 1980, p. 258
Microcomputers in the Training Lab
Tinker, 1980, p. 250

COMPUTER LITERACY

A Byte of BASIC
Hopper, 1980, p. 62
A Case for Information Literacy
Schimming, 1980, p. 58
College Computer Literacy
Wright, Keller, Barnes, Garson and Miles, 1981, p. 311
Computer Literacy for Elementary School Teachers
LaFrance, 1981, p. 80

Computer Literacy for Liberal Arts Students: *The Case of* Computer Mapping
Clark, 1979, p. 107
A Computer Workshop for Elementary and Secondary Teachers
Dersham and Whittle, 1980, p. 65
Developing Computer Literacy at a Liberal Arts College
Kolodny and Ott, 1981, p. 59
Integrating Computer Literacy into Existing Curriculum: Some Case Studies
Heller and Martin, 1981, p. 74
Microcomputers and Computer Literacy: A Case Study
Ellison, 1980, p. 68
A Novel Approach to Computer Literacy for Natural Science Students
Rodriguez and Anger, 1981, p. 66
Precollege Computer Literacy
Anderson and Watchel, 1981, p. 240

COMPUTER SCIENCE

The C-Basic Intepreter: A Learning Tool for Compiler Construction
Rohr, 1981, p. 161
Computer Courses at Hong Kong Polytechnic
Hickerson and Mead, 1982, p. 24
Computer Curricula in Evolution
Kroeber, 1982, p. 15
Computer Literacy 1982: An Update
Hertlein, 1982, p. 238
Computer Managed Instruction in the Field of Computer Science
Bulinski, 1982, p. 28
A Computer Software Technician Program at Portland Community College
Hata, 1980, p. 287
Computer Support for Courses in Data Entry and Word Proce-sing
Sunkel, Rickman and Hobbs, 1979, p. 205
Cognition and Programming: Why Your Students Write Those Crazy Programs
Soloway, Bonar, Woolf, Barth, Rubin and Ehrlich, 1981, p. 206
Data Structures at the Associate Degree Level
Dempsey, 1980, p. 152
A Descriptive Approach to the Teaching of Computer Organization
Vranesic, 1979, p. 101
Development of a Resource Center for Introductory Computer Science
Lee, 1979, p. 135
Development of Communication Skills in Software Engineering
Beidler and Meinke, 1980, p. 143
Firmware Development
Cook, 1981, p. 198
Initial Evaluation Results for an Introductory Programming Coursew tihout Lectures
Hazen, Daly, Embly, Nagy, and Prange, 1979, p. 150
An Interactive, Instructional, CODASYL Data Base Management System for Main Frames and Micros
Niccolai and Lewis, 1982, p. 186
A Liberal Arts Computer Literacy Course: A Preliminary Report
Franklin, 1982, p. 120
A More Analytical Approach to Teaching the Introductory Programming Course
Deimel and Moffat, 1982, p. 114
Pre-Assembler: A Bridge (Course) Over Troubled Waters
Turner and Keefer, 1981, p. 127
Opal -- A Hardware Independent Assembly Language for Education
Luce, 1981, p. 121
Rating the Major Computing Periodicals on Readability
Lemos, 1982, p. 170
Required Freshman Computer Education in a Liberal Arts College
Wetmore, 1980, p. 139
A Secondary Level Curriculum in System Dynamics
Roberts and Deal, 1980, p. 281

Simulation of Computer Architecture Using a Hierarchical Computer Hardware Description Language
    Skelton and Walker, 1981, p. 115
Small One User Systems Versus Time Sharing Systems in the Introductory Course: A Controlled Experiment
    Shub, 1982, p. 123
SM 1802 A Microcomputer Simulator
    Schmiedekamp, 1982, p. 216
Software Documentation for Student Projects
    Rohr, 1981, p. 195
A Special Computer for Teaching How Computers Work
    Monk, 1982, p. 245
A Successful Experience with Programming Language at a Liberal Arts College
    Cleveland, 1981, p. 154
Systematic Assessment of Programming Assignments
    Bishop, 1980, p. 147
Teaching Fundamental Technology by Subject Matter Professionals Using Programmable Calculators
    Taylor and Woolverton, 1981, p. 169
Using Robots to Help Teach Assembly Language
    Scott, 1982, p. 192
What Programming Language, If Any, for Beginners?
    Hughes, 1979, p. 141

COMPUTER SERVICES

Development of an Academic Software System in a Small College Environment
    Moberg, Verkade and Dahl, 1982, p. 65
The Development of Library Automation in a Medium Size Academic Library: The U.A.B. Experience
    Stephens, 1982, p. 101
A Discussion and Real-Time Demonstration of Valuable Programming Features
    Cornelius, 1982, p. 95
Electronic Mail and Computer Based Conferencing
    Heller and Hogan, 1982, p. 234
Evaluation and Selecting Computer Studies Textbooks: Arts, Science, or Chance?
    Mitchell, 1981, p. 111
Funding of Videodisc Projects
    Breitler, 1981, p. 34
Goverance and Policy Regarding Computer Services and Information Systems in Small Two Year Colleges
    Conrad, 1982, p. 59
Programming Project Management
    Rodriguez, Acuna, Anger and Babylonia, 1982, p. 380
Linking Microcomputers to Timesharing Systems: Efficient Computing in a Liberal Arts Environment
    Steen, 1982, p. 69
A Micro Lab: Mixing Apples and Oranges
    Dooley and Traux, 1982, p. 97

EDUCATION

Computing as a way of Brainstorming in English Composition
    Burns, 1981, p. 105
Continuing Education
    Settle, Pleva, Weiler, Mozes and Alley, 1981, p. 152
The Future of Microcomputers in Education: A Vendor's View
    Polin, 1981, p. 243
Network Information Resources for Computers in Teaching and Learning
    Zinn, 1981, p. 239
The Role of a Computer-Based University-Wide Testing Service to Manage Large Enrollment Courses
    Wood and Ortega, 1981, p. 94

A Study of the Effectiveness of Computer-Assisted Feedback for Modifying Teacher Behavior in an Inservice Setting
Hasselbring and Crossland, 1981, p. 99

ELEMENTARY SCHOOL COMPUTING PROGRAMS

Case Studies of Children Learning LOGO
Burns, 1982, p. 268
A Comparison of the Problem-Solving Styles of Two Students Demonstration, Field Test Results and Discussion of Computer Software Based on Video Game Formats
Ruggles and Chafin, 1982, p. 269
Computer-Assisted Instruction in Reading Used for Kindergartners and First Graders in Dillon, Mont. 79-80
Bandelier, 1981, p. 269
Computer Literacy for the Classroom Teacher, K-8
Heller and Martin, 1982, p. 39
Computers and the Nursery School
Swigger and Cambell, 1981, p. 264
Computing Concepts in Elementary Schools
Christensen, 1981, p. 35
Elementary School Applications: LOGO
Lola, Lewis, Overall, Gorman, Murphy, and Watt, 1981, p. 81
A Humane, Forgiving System for Beginning Programmers in BASIC
Kushan, 1982, p. 269
Learning LOGO: Computer Language for Children
Watt, 1979, p. 255
Language in the Logo Computer Culture
Solomon, 1979, p. 250
Matching Computer Based Reading Instruction to Preschool Gifted Childrens Modality Strengths: A Whole Brained Approach to Learning
Krause, 1982, p. 269
OZNAKI and BEYOND
Cohen, 1979, p. 170
Reading Skills Courseware Series
Belton, 1982, p. 268
The Role of the Classroom Teacher
Cook, 1982, p. 268
The Scarsdale Project: Integrating Computing into K-12 Curriculum
Sobol and Taylor, 1980, p. 155
A Study of Preschool Children's Use of Computer Programs
Lewis, 1981, p. 272
Teaching LOGO in Child Centered Classrooms
Dubitsky, 1982, p. 269

ENGINEERING

A Case Study of an External Evaluation Mechanism
Moeller and Niederjohn, 1982, p. 138
Computer-Augmented Video Education in Electrical Engineering at the U.S. Naval Academy
Lim, Hagee, and Pollak, 1980, p. 90
Computers: New Avenues for Engineering Students at the Community College
Brillhart and Shawn, 1979, p. 21
Facilitating Small System Modular Instruction with an Information System
Evans and Himmelblau, 1981, p. 332
Interactive Microcomputers with Graphics Provide Teaching Assistance in Engineering and Technology
Moore and Baxter, 1982, p. 133
On Introducing Computer-Aided Engineering Techniques into an Established Mechanical Design Curriculum
Sutherland and John, 1979, p. 18
Low Level Computer-Assisted Design on an Apple II with Pascal
Deal, 1982, p. 129

Microprocessors in the Whole of Engineering Education: The Contemporary Micro Lab
Brubaker, Cain, Gowdy and Krutz , 1982, p. 105
Personal Computer Simulation Programs as Teaching Aids: A Successful Experiment in Undergraduate Electrical Engineering Education
de Geus, Pucacco and Roher, 1981, p. 336
Switching over to the Micros
Maurer, 1981, p. 327

FACULTY CONSIDERATIONS

Computer Training for the Faculty at Drew University
Metzler, Nelson and Carroll, 1979, p. 378
Faculty Training and Development in Academic Computing
Gordon, 1982, p. 379
Transitioning Instructor Roles in Computer-Based Environments: A Preliminary Program for Improving Instructor Attitudes
Dobrovolny, McCombs and Judd, 1979, p. 380

GRAPHICS

Computer Graphics as a Lecture Aid in Undergraduate Mathematics
Porter, 1979, p. 237
Graphic Design Issues in Computer-Based Education
Nicholas, Weissman, and Shankman, 1981, p. 109
Graphics and Computer-Based Learning
Herman, 1981, p. 150
Graphics in Computer Science
Moore, 1981, p. 287
The Human Factors of Color Display-Based Instruction
Durrett, Zwiener, and Fruend, 1981, p. 284
Some Uses of Computer Graphics in the Calculus Classroom
Maurer and Donovan, 1981, p. 281
Tessellations and the Teltronix 4051
Tanis, 1979, p. 233
Using Graphics for CAI with a Microcomputer
Ward and Rudnick, 1979, p. 271

HANDICAPPED

A Computer-Assisted Instruction System for the Blind and Visually Impaired
Ballenger, 1979, p. 274
A Microcomputer/Videodisc CAI System for the Moderately Mentally Retarded
Thorkildsen, Bickel, and Williams, 1979, p. 285
Personal Computing to Aid the Handicapped: The Johns Hopkins First Nation Search
Hazan, 1981, p. 244
SPEDINFO: A Decision Making Support Feature for the PLATO System
Cronin and Rizza, 1979, p. 280

HEALTH CARE

An Adverse Drug Interaction System: Its Design, Operation and Use
Jaufmann, Williams and Barletta, 1979, p. 354
Computers and Continuing Health Education
Pogue, Comer, Eisele, Peterson and Held, 1981, p. 149
Computer Literacy: Practical Necessity
Jackson and McKee, 1982, p. 155
Design of a CAI Tool to Generatively Teach Etymology
Garcia and Rude, 1979, p. 371
Developing a Teaching/Learning Experience for Nurses in Fundamentals of Computer Programming Preliminary to Nursing Research
Erat and McGrath, 1979, p. 316

An Educational Program in Medical Computing for Clinicians and Health Scientists
Hybl and Reggia, 1980, p. 276
Education for Careers in Health Computing
Duncan, Austin, and Pogue, 1982, p. 311
Educators for Computing and Health Care - A New Profession
Duncan, Katz, and Pogue, 1982, p. 345
A Problem Solving Model in Continuing Education for Dental Practitioners
Lipson, Keith and Jones, 1979, p. 301
Toward the Development of a Computer-Managed Instructional System in Pharmacy, Based Upon Integrated Database System Concepts
Henkel and Williamson, 1979, p. 360
A Training Program in Health Computer Sciences
Ellis and Gatewood, 1979, p. 311

HUMANITIES

Computer Assistance in the German Individualized Instruction Program at the Ohio State University
Taylor, 1981, p. 132
Computer-Assisted Instruction in Melodic Composition
Dworak, 1981, p. 136
Computer-Assisted Instruction in Music: Analysis of Student Performance 1973-80
Kuhn and Lorton, 1981, p. 142
Creativity through the Microcomputer
Bass, 1980, p. 38
The Effect of Technology in the Traditional Arts: Comparison and Contrast of Female and Male Students' Attitudes toward Ear-Training Computer-Assisted Instruction
Dillard and Killam, 1982, p. 141
From a Theory of Reading to Practice via the Computer
Johnson and Baldwin, 1980, p. 46
Generative CAI for Foreign Languages
Decker and Rice, 1979, p. 326
Giving Advice with a Computer
Garson, 1980, p. 42
Melodic Perception Development and Measurement through Computer-Assisted Instruction
Ray and Killman, 1979, p. 329
Non-Harmony: A Vital Element of Ear Training in Music CAI
Groom-Thornton and Corbet, 1980, p. 54
The Stanford Logic Course: Design, Dissemination and Demonstration
Ager and McDonald, 1979, p. 335

MATHEMATICS

Applications of the Definite Integral and Basic
Lindstrom, 1981, p. 313
Calculus Tutorial System
Arganski, 1982, p. 381
College Mathematics
Maurer, Donovan, Mackey, Lindsay, Harp, Walgren and Morrow, 1981, p. 341
Computer-Aided Education in Calculus -- Limits
Hoffman, 1982, p. 382
Computer Applications in a Finite Mathematics Course
Piegari, Abernathy, and Thorsen, 1980, p. 179
A Computer-Assisted Course in Biomathematics
Wong, 1980, p. 184
A Computer-Based Dialogue for Developing Mathematical Reasoning of Young Adolescents
Trowbridge and Bork, 1981, p. 324
Computer Graphics as a Lecture Aid in Undergraduate Mathematics
Porter, 1979, p. 237
Computer Symbolic Math
Stoutemeyer, 1980, p. 194
The Effectiveness of Mathematics Instruction in the CAI Mode: A Comparative Study
Pulat and Wang, 1982, p. 258

An Exmination of the Use of Computers to Increase Student Motivation and Understanding in Relation to Complex Mathematical Models
Dennis, Stevenson, and Colella, 1979, p. 194
FORTRAN and Introductory Numerical Method: A Freshman Course
Borse and Clump, 1982, p. 196
FUNGRAPH: A Language for Computer Inspired Learning
Miller, 1982, p. 382
Helping Teachers Teach Elementary Mathematics with Microcomputers, Calculators and Manipulatives
Moore, 1982, p. 254
Laboratory Integration in Finite Math and Survey Calculus
Schoenefeld, 1982, p. 383
A Method for Experimenting with Calculus Using CAI
Anger and Rodriquez, 1980, p. 171
Microcomputers in a Two Year C.L.I.M.B. Center (Center for Learning and Instruction in Mathematics - Concept Building
Thompson, 1982, p.
Natural Language Computing in a Linear Algebra Course
Geist, Kraines and Fink, 1982, p. 203
Programmable Calculators in the MSE Curriculum: A Report on Experiences in the Trenches
McNulty and Palmer, 1982, p. 248
Starting a Computer-Based Learning Project
MacKichan, Adams and Hunter, 1981, p. 320
Teaching Experimental Design with a Mathematical Model Algorithm
Ward, 1981, p. 8
Tessellations and the Teletronix 4051
Tanis, 1979, p. 233
An Undergraduate Curriculum in Computer Graphics
Prigger, 1982, p. 383
The Use of Computers in a First Course in Numberical Analysis
Motteler, 1982, p. 209
Use of Computers in Mathematics Instruction: Getting Started
Gordon, Greenwell, Hale and Rice, 1982, p. 75

MICROS AND MINIS

The AVID System: A Multiple-User Minicomputer-Based System for Computer-Based Education
Scholz and Sloane, 1979, p. 264
Computer-Based Instruction for the Public Schools: A Suitable Task for Microprocessors?
Taylor, 1980, p. 233
Considerations and Guidelines for ZDeveloping Basic Skills Curriculum for Use with Microcomputer Technology
Caldwell, 1980, p. 31
Creativity Through the Microcomputer
Bass, 1980, p. 38
Guidelines for the Successful Selection and Operation of Mini and Micro Computers
Gale, 1981, p. 277
The ISU Microsupport Net: A Prototype System for Personal Computing in Education
Horton, 1979, p. 268
Microcomputers and Computer Literacy: A Case Study
Ellison, 1980, p. 68
Microcomputers as Laboratory Instruments: Two Applications in Neurobiology
Olivo, 1980, p. 81
Microcomputer-Assisted Study and Testing System
Garraway, 1980, p. 200
Microcomputer-Generated Videotapes
Cooke and Martin, 1979, p. 347
Microcomputer Networks
Tinker and Pogue, 1981, p. 309
Microcomputer Software Development: New Strategies for a New Technology
Kehrberg, 1979, p. 343

Microcomputers in the Teaching Lab
Tinker, 1980, p. 250
Using Graphics for CAI with a Minicomputer
Ward and Rudnick, 1979, p. 271

MINORITY INSTITUTIONS

Academic Computing: A Sample of Approaches in Minority Institutions
Marshall, 1980, p. 238
Computer Enhancement of Cultural Transition
Gauss, 1979, p. 60
Computer Use in Chemistry at a Minority Institution
Beck, 1980, p. 245
Computing in Minority Institutions: 1976-1977
Marshall, Lewis, Jones, Hamblen, 1979, p. 65
Educational Use of Computers in Puerto Rico
Anger, 1980, p. 249

PEDAGOGY

Computer-Assisted Test Construction via Automatic Program Generation: Using PROBGEN II to Create Individualized Exams and Problem Sets
Collins and Duff, 1979, p. 114
Computer Dialogs to Aid Formal Reasoning
Arons and Bork, 1979, p. 130

PRE-COLLEGE COMPUTER SCIENCE

Algorithmics in High School
Gruska, 1982, p. 335
A COMAL System for Irish Secondary Schools
Ryan, 1982, p. 373
Computer Education at Moreland Elementary School: The First Year
Lazerick, 1982, p. 369
Computer Literacy in the Elementary School: A District Wide Program
Goodson, 1982, p. 235
The Duke University Computer Kamp: "DUCK"
Bowyer, Ray and Laxer, 1982, p. 364
An Original High School Programming Course that Works
Shirley, 1982, p. 341
Problem Solving, Teamwork and Structural Programming in High School" The Story of High School Computer Science Project
Baird, 1982, p. 331

PRE-COLLEGE ENVIRONMENT

A Computer Loan Service
Dietz and Kemper, 1979, p. 161
Management of Instructional Computing: Advice from Ten Pre-College Institutions
Hunter and Hargan, 1979, p. 164
OZNAKI and BEYOND
Cohen, 1979, p. 170

PRE-COLLEGE INSTRUCTIONAL NON-COMPUTER SCIENCE

Computer-Based Instruction for the Public Schools: A Suitable Task for Microprocessors?
Taylor, 1980, p. 223
Computerized Individual Instructional Program
Caccamo and Watkins, 1982, p. 272
Microcomputers and Maxicomputers: Can we Coexist in Educational Computing
Finkel, 1982, p. 279

Microcomputer/Videodisc CAI -- Some Development Consideration
    Tornkildsen and Atlard, 1980, p. 230
The Non-Technical Factors in the Development of CAI
    Mocciola, 1980, p. 236
Organization Support for Professionals Interested in Precollege Education
    Ricketts, Adams, Goodson, Noursund, Smith and Spuck, 1982, p. 346
A Report on Computing in Iowa High Schools
    Sjoredsma, 1982, p. 285

PRE-COLLEGE SERVICES

Micro-Computers and Maxi-Computers: Can we Coexist in Educational Computing?
    Finkel, 1982, p. 279

PRE-COLLEGE TEACHER EDUCATION

ACM Elementary and Secondary Schools Subcommittee Progress Report
    Moursand, 1980, p. 125
An analysis of Computer Education Needs for K-12 Teachers
    Milner, 1979, p. 27
A Computer Education Course for Elementary Teachers
    East, 1982, p. 347
A Computer Workshop for Elementary and Secondary Teachers
    Dersham and Whittle, 1980, p. 65
Computing Competencies for School Teachers
    Taylor, Poirot and Powell, 1980, p. 130
Computing Competencies for School Teachers: A Preliminary Projection for all but Teachers of Computing
    Taylor, Hamblin, Powell and Poirot, 1979, p. 39
Computing Competencies for School Teachers: A Preliminary Report for the Teachers of Computing
    Taylor, Hamblin, Powell and Poirot, 1979, p. 36
AnInductive Approach to Learning CAI Programming in BASIC
    Ragan and Dillingham, 1982, p. 348
A Masters and CAGS Degree Program: Computers in Education
    Roberts, 1982, p. 349
Microcomputer Usage for Special School Teachers and Counselors: Development of a Symposium
    Rhoads, 1982, p. 348
Overview of Two Mini Courses and Two Conferences in Educational Applications of Microcomputers for In Service Teachers
    Underhill, 1982, p. 349
Stages of Development in Introducing Computing to Teachers
    Dennis, 1979, p. 31
Topics and Programs for Elementary and Secondary School Administrators
    Mins, 1982, p. 347

PHYSICS

Interfacing with ATARI 800 Paddle Ports
    Williams and McInerney, 1982, p. 225
Introductory Quantum Mechanics Using Microcomputers
    Min and Kosterman, 1982, p. 213
SM 1802 A Microcomputer Simulator
    Schmiekamp, 1982, p. 216

SCIENCE

Classical Mechanics with Computer Assistance
    Davis, 1980, p. 86
Computer-Assisted Instruction in Secondary Physical Science: An Evaluation of Student Achievement and Attitudes
    Marks and Bartholomew, 1981, p. 228

Demographic Techniques in Ecology: Computer-Enhanced Learning
Gatz, 1980, p. 76
Design of Computer-Based Instructional Materials to Enhance the Problem-Solving Abilities of Science Students
Suits and Lagowski, 1981, p. 184
An Educational Computer Network for the Users of Chemical Instrumentation
Pleva and Settle, 1981, p. 175
Five Applications of Computers in Chemical Education
Johnson, 1979, p. 92
Microcomputers as Laboratory Instruments: Two Applications in Neurobiology
Olivo, 1980, p. 81
Newton -- A Mechanical World
Bork, Franklin, Katz and McNelly, 1981, p. 179
Quantum Chemistry Instructional Package Using Interactive FORTRAN
Joshi and Eilers, 1979, p. 86
Science Literacy in the Public Library -- Batteries and Bulbs
Arons, Bork, Franklin, Krutz and Collea, 1981, p. 220
Teaching Simulation Techniques with Microcomputers
Spain, 1981, p. 225

SOCIAL SCIENCE

A Computerized Course in Elementary Statistics: Educational Objectives and Methods
Evans, 1981, p. 254
An Empirical Analysis of "Learnit" A Microcomputer Program for Facilitating Memorization
Schilmoeller, Schilmoeller and King, 1982, p. 306
Games People will Play: Development of a Method to Assess Interest in Instructional Games
Neideffer and Evans, 1981, p. 259
STATAN, A Query Language to Help Students Practice Statistics
Parkhurst, 1982, p. 302
The Study of U.S. Regional Shifts, Metropolitan Growth and Neighborhood Structure with Census Microdata
Benenson and Just, 1981, p. 245
Teaching Demography Through the Computer
Stack, 1979, p. 211
Teaching Substantive Sociology Through Computer Assisted Homework: The Development of CLEAR
Conklin, 1982, p. 289
The Use of Census Data Analysis to Enhance the Sociology Curriculum
Benenson and Just, 1979, p. 297

STRUCTURED PROGRAMMING

FORTRAN 77: Impact on Introductory Courses in Programming using FORTRAN
Friedman, 1980, p. 103
Shall We Teach Structured Programming to Children?
LaFrance, 1980, p. 261
Structured Gaming: Play and Work in High School Computer Science
Moshell, Amann and Baird, 1980, p. 266
Structured Machine Language: An Introduction to Both Low- and High-Level Programming
Hannay, 1980, p. 119
Structured Systems Analysis Tutorial
Owles and Powers, 1981, p. 112
The Use of Programming Methodology in Introductory Computer Science Courses
Alpert, 1980, p. 96
Using Model Based Instruction to Teach PASCAL
Czejdo, 1980, p. 112

SOFTWARE DEVELOPMENT

ADA Tutorial
Mathis, 1981, p. 310

A CAI Authoring Environment for Non-Programmers
Rinehart and Rosenthal, 1982, p. 108
A CAI Program for Literary Analysis: MARSY/ERB
Schwartz, 1982, p. 108
Development of the Video-Computer Courseware Implementation System at the University of Utah
Brandt and Knapp, 1982, p. 107
Educational Software: Its Creation and Evaluation
Johnson, 1982, p. 113
Program Development Methodology
Turner, 1981, p. 36
Voice Controlled CAI
Horn, 1982, p. 108
ZGRASS: A Simple Language for Graphics Courseware
Garson, 1982, p. 107

TEACHING TRAINING

Alternative Teaching Training: A Possible Solution to the Crisis in Computer Science Education
Berghel, Dale, Etlinger, Franklin, Rice and Seef, 1982, p. 263
Computer Literacy for Teachers
Hopper and Brackhart, 1982, p. 323
Computer Literacy for Teachers: An Intensive Program
Vockell and Rivers, 1982, p. 326
Computers in Public School Classrooms: Teacher Training Issues and Concerns
Pratcher, Billings, Culp, Leuhrmann and Poirot, 1982, p. 312
Developing a Computer Curriculum for a College of Education
Vinsonhaler and Wolverton, 1982, p. 161
Computers in Education: What's an Appropriate Way to Teach Educators about the Uses of Computers?
Fetcher, 1982, p. 318
Educating Urban Elementary Teachers in Computer Science
Carroll and Johnson, 1981, p. 302
Helping Teachers Teach Elementary Mathematics with Microcomputers, Calculators and Manipulatives
Moore, 1982, p. 254
Microcomputer Classroom Use Patterns
Judd, 1981, p. 297
Microcomputers in Education: A Course in Computer Literacy for Educators
Anderson, 1981, p. 292
Microcomputer Education for Teachers: Threats, Techniques and Triumphs
Nickles, Adams and Culp, 1982, p. 264
Summer Institutes in Microcomputing for Secondary Teachers
Baker, 1982, p. 351
A Teachers Introduction to Computers
Taffe and Weissmann, 1982, p. 359
Teacher Training Programs: Use of Microcomputers
Coltharp, Beougher, Corbitt, Driver and Rezac, 1982, p. 230
Quality Inservice Training: The Key to Establishing and Maintaining a Computer Education Program
Shotwell, 1982, p. 355

TESTING/PLACEMENT

Computer Managed Placement in Mathematics Instruction for Health Occupation Students
Boyle and Magnant, 1980, p. 214
Microcomputer Assisted Study and Testing System
Garraway, 1980, p. 200
RIBYT -- A Database System for Formal Testing and Self-Assessment
Frhs, 1980, p. 205

TOOLS AND TECHNIQUES OF INSTRUCTION

Considerations and Guidelines for Developing Basic Skills Curriculum for Use with Microcomputer Technology
Caldwell, 1980, p. 31

A Dynamic Process in Teaching Techniques
Effarah, 1980, p. 25

Hypetext: A General Purpose Educational Computer Tool
Ward and Bush, 1980, p. 19

TOPICS

ADA Programming Language
Mathis, 1982, p. 317

Developing a computer Curriculum for a College of Education
Vinsonhaler and Wolverton, 1982, p. 161

Computer Literacy: Practical Necessity
Jackson and McKee , 1982, p. 155

The Effect of Technology in the Traditional Arts: Comparison and Contrast of Female and Male Students' Toward Ear-Training Computer-Assisted Instruction
Dillard and Killam, 1982, p. 141

Instructional MIS: User and Management Considerations
Avery, 1982, p. 229

MicroSIFT Project and Courseware Evaluation
Plummer, 1981, p. 190

The Next Step: The Use of Micosynergistic (Microcomputer Cluster) in Education
Motto, 1981, p. 24

An Overview of Micro-CMI
McIsaac, 1981, p. 238

Pascal Tutorial
Haiduk, 1981, p. 192

The Possible Effect of Color on Computer-Assisted Learning
Nickles, 1981, p. 191

Projects Supported Through National Networking
Heller, 1981, p. 276

Research and Application of Videodiscs
Pollak, 1981, p. 78

Transportability of Instructional Computer Programs: Issues and Examples
Herman, 1981, p. 18

UNDERGRADUATE NURSING CURRICULUM USE OF COMPUTERS

An Authoring System for Developing Computer-Assisted Instructional Programs on Nursing Process
Grobe, 1982, p. 265

Course Description for EDUC 872, Readings in Education
Neil, 1982, p. 265

An Orientation to Computer Literacy in Nursing
Thomas, 1982, p. 266

The Role of the CAI Author as Curriculum Developer. What Instructional Design Theory can Tell us About Planning CAI
Tarp, 1982, p. 266